



## 저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Ph.D. DISSERTATION

Reactive Collision Avoidance  
Using the Velocity Obstacles Concept  
in Polar Coordinates

극 좌표계에서 Velocity Obstacles 개념을  
사용한 반응적 충돌 회피 기술

BY

TAESEOK LEE

FEBRUARY 2014

DEPARTMENT OF ELECTRICAL ENGINEERING AND  
COMPUTER SCIENCE  
COLLEGE OF ENGINEERING  
SEOUL NATIONAL UNIVERSITY

# Abstracts

In order to autonomously navigate in an unknown environment, a mobile robot should perceive the environments precisely and generate fast-moving path without collisions. In recent years, as the operating environment is becoming more and more complicated, considering various factors such as multiple agents, moving obstacles becomes an important issue in autonomous navigation. Therefore, it is necessary to develop a collision avoidance navigation algorithm which is effective in a variety of situations.

A centralized navigation system collects information of the environments and all robots, and decides trajectories of each robot. As the environment gets more complex, calculating the collision-free trajectory is difficult. A distributed navigation system which controls the robot individually cannot guarantee optimal path of the robot, but it is easy to apply depending on the situation. This dissertation addresses local and reactive navigation without centralized coordination or control.

A velocity obstacle approach one of local and reactive navigation methods is re-analyzed here. Most of the conventional velocity obstacle approaches are analyzed in Cartesian coordinates. The proposed approach of this dissertation performs collision prediction and avoidance motion planning of a mobile

robot with non-linear velocity based on robot-centered polar coordinates. By re-analyzing the velocity obstacles concept in robot-centered polar coordinates, obstacle avoidance process has been simplified.

Depending on the direction of the robot and the moving obstacles, the robot occasionally selects oscillating velocity as a result of using the conventional velocity obstacle approaches. In order to overcome the oscillation, new strategy which decides velocity of the robot to avoid collision with the oscillation-free path is designed. The proposed evaluation function is containing the current status of the robot, the relation between the robot and the obstacle, and the distance to the destination. The evaluation function is used for the robot velocity decision.

Numerous simulations have been implemented to validate the proposed approach as well as the conventional algorithms. The performance of the proposed approach is verified by comparing the traveling time, distance, and computation time, and the smoothness of the robot path with the conventional algorithms.

**Key words:** Collision avoidance, Velocity obstacles, Polar coordinates, Multiple agents, Motion planning

**Student Number:** 2009-30205

# Contents

<b>Abstract</b>	<b>i</b>
<b>Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>viii</b>
<b>Nomenclature</b>	<b>ix</b>

<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 Background and motivation .....	1
1.2 Related works .....	4
1.3 Contribution .....	9
1.4 Organization .....	13
<b>Chapter 2 Velocity Obstacle Approaches</b>	<b>14</b>
2.1 Velocity Obstacles .....	14
2.2 Dynamic Velocity Space .....	17

<b>Chapter 3 Problem Description of Reactive Control</b>	<b>23</b>
3.1 Reactive Control in Cartesian and Polar Coordinates .....	23
3.2 Basic Assumptions and Definitions .....	28
<b>Chapter 4 Velocity Obstacles in Polar Coordinates</b>	<b>32</b>
4.1 Robot-centered Polar Coordinate Representation of an Obstacle ...	32
4.2 Desired Velocity Generation .....	34
4.3 Velocity Obstacle Region Generation .....	36
4.4 New Velocity Decision .....	42
4.4.1 Basic Method of Choosing Alternative Velocity .....	42
4.4.2 Proposed Method: Evaluating Two Alternatives .....	45
4.5 Kinematic Constraints and Execution .....	49
<b>Chapter 5 Analysis of Velocity Obstacles in Polar Coordinates</b>	<b>52</b>
5.1 Collision-free Navigation .....	52
5.2 Smoothness of the Robot Trajectory .....	60
5.3 Local Minima Avoidance .....	64
<b>Chapter 6 Simulation Results</b>	<b>73</b>
6.1 Implementation Setups .....	73
6.2 Simulations with a Single Dynamic Obstacle .....	75
6.2.1 Scenario 1: An Obstacle with Zero Angular Velocity .....	76
6.2.2 Scenario 2: An Obstacle with Non-Zero Angular Velocity .....	83

6.3 Simulations with Multiple Dynamic Obstacles .....	86
6.3.1 Scenario 3: Four Moving Obstacles .....	89
6.3.2 Scenario 4: Ten Moving Obstacles .....	94
6.3.3 Scenario 5: Seven Moving Obstacles in a Circle .....	99
6.3.4 Scenario 6: Five Moving Obstacles with Velocity Changes ..	102
<b>Chapter 7 Conclusions</b>	<b>106</b>
<b>Bibliography</b> .....	<b>111</b>

# List of Figures

1.1 Obstacle avoidance scheme of the dynamic velocity space algorithm ···	6
2.1 A graphical procedure of the original velocity obstacle algorithm ·····	15
2.2 A graphical procedure of the dynamic velocity space algorithm ·····	18
2.3 The setpoint velocity generation of the dynamic velocity space ·····	21
3.1 Motion planning process in global Cartesian coordinates ·········	25
3.2 Motion planning process in robot-centered polar coordinates ·······	27
4.1 Representation of the moving robot and obstacle ···········	33
4.2 The trajectory of the robot from the current position to the goal ·····	35
4.3 The situation of that the robot arrives at the center of the obstacle at $\kappa$ ·	37
4.4 The two end boundaries of the robot trajectory ···········	39
4.5 Robot velocities toward the left and right boundaries of the obstacle ···	41
4.6 An example of drastic turning to avoid urgent collisions ·········	43
4.7 An example of oscillation ···········	44



4.8 The concept of selecting two alternative velocities .....	46
5.1 The crossing points of the robot trajectory and the obstacle boundary ..	53
5.2 The robot trajectories passing both ends of the obstacle boundary .....	54
5.3 The velocity obstacle region at time $\kappa$ on polar coordinates .....	56
5.4 The velocity obstacle region for the obstacle B on polar coordinates ..	57
5.5 The velocity obstacle region and the left and right end boundaries .....	58
5.6 The robot trajectories passing the obstacle boundary on the right side ..	59
5.7 Examples of geometric continuity of a two-dimensional curve .....	62
5.8 The velocity selection results causing oscillations .....	63
5.9 The robot trajectory with oscillation and the local minima problem.....	67
5.10 The stuck situation due to the local minima problem.....	68
5.11 The evaluation factors and corresponding evaluation function terms ..	69
6.1 The two-wheeled differential drive robot model.....	74
6.2 The global map and the robot trajectories of the first scenario .....	77
6.3 The velocity obstacle regions of three obstacle avoidance algorithms ..	79
6.4 The trajectory of the robot using the proposed algorithm .....	80
6.5 The trajectory of the robot using the original velocity obstacles .....	81
6.6 The trajectory of the robot using the dynamic velocity space .....	82
6.7 The robot trajectories of the second scenario .....	84

6.8 The velocity obstacle regions of the conventional and new algorithms ·	85
6.9 The global map and the robot trajectories of the third scenario ······	90
6.10 The velocity obstacle regions of three collision avoidance algorithms ·	92
6.11 The global map and the robot trajectories of the fourth scenario ······	95
6.12 The velocity obstacle regions of the proposed navigation algorithm···	97
6.13 The global map and the robot trajectories of the fifth scenario ······	100
6.14 The global map and the robot trajectories of the sixth scenario·····	103

# List of Tables

1.1. Properties of reactive obstacle avoidance algorithms .....	9
4.1 Algorithm of the velocity obstacles in polar coordinates .....	51
5.1 Analysis of the proposed evaluation function .....	71
6.1 Initial information for scenario 1 and 2 .....	75
6.2 Comparison of the simulation results of scenario 1 .....	80
6.3 Comparison of the simulation results of scenario 2 .....	85
6.4 Initial information for scenario 3 and 4 .....	87
6.5 Initial information for scenario 5 and 6 .....	88
6.6 Comparison of the simulation results of scenario 3 .....	93
6.7 Comparison of the simulation results of scenario 4 .....	98
6.8 Comparison of the simulation results of scenario 5 .....	102
6.9 Comparison of the simulation results of scenario 6 .....	105
7.1 Overall comparison of the simulation results .....	109

# Nomenclature

Notation	Description	Chapter
$\mathbf{p}_A^{Cart}$	The position of an agent(robot or obstacle) A in Cartesian coordinates	2,3,4,6
$\mathbf{p}_{BA}^{Cart}$	The relative position of an agent B with respect to an agent A in Cartesian coordinates	2,4,6
$R_A$	The radius of an agent A	2,3,6
$R_{BA}$	The sum of the radii of two agents A and B	2,3
$\mathbf{v}_A^{Cart}$	The velocity of an agent A using Cartesian coordinated representation	2,3
$\mathbf{v}_{AB}^{Cart}$	The relative velocity of an agent A with respect to an agent B in Cartesian representation	2
$VO_{AB}^\kappa$	The velocity obstacle region of an agent A with respect to B by the original velocity obstacles before time $\kappa$	2
$r_j$	The $j$ -th specific path of the robot for the dynamic velocity space algorithm	2
$\mathbf{p}_{cij}$	The point intersecting between $r_j$ and the collision band $i$ -th boundary	2
$\mathbf{v}_{ij}$	The velocity on the boundary of DOV set for $r_i$ using polar coordinated representation (lower: $i=1$ , upper: $i=2$ )	2

$\mathbf{v}_{des}$	The robot velocity towards the goal point using polar coordinated representation	2,3,4
$\mathbf{v}_{SP}$	The velocity with the largest linear velocity among the lower limits of DOV set along $\mathbf{v}_G$	2
$\mathbf{v}_{MT}$	The nearest accessible velocity from $\mathbf{v}_G$ or $\mathbf{v}_{SP}$	2
$\mathbf{v}_d$	The setpoint velocity	2
$\mathbf{p}_A^{polar}(\rho_A, \phi_A)$	The position of an agent A in polar coordinates	3
$\mathbf{p}_{BA}^{polar,rel}$	The relative position of an agent B with respect to A in polar coordinates centered at A	3,4,
$\mathbf{v}_A(\dot{r}_A, \dot{\theta}_A)$	The velocity of an agent A using polar coordinated representation	3,4,5,6
$r_{range}$	The maximum sensing range	3,5
$\mathbf{O}_{rot}$	The center of robot's circular trajectory	4
$R_{BA}^{rot}$	The radius of rotation of an agent A to reach the B	4,5
$\mathbf{v}_{BA}^{VO}$	The velocity of an agent A to reach the B with maintaining $R_{BA}^{rot}$	4
$\mathbf{p}_{BA}^{VRB}$	The contact point between the robot boundary and the contacted right end robot trajectory	4,5
$\mathbf{p}_{BA}^{VLB}$	The contact point between the robot boundary and the contacted left end robot trajectory	4,5
$\mathbf{v}_{BA}^{VRB}$	The velocity for the $\mathbf{p}_{BA}^{VRB}$ at the particular time	4,5
$\mathbf{v}_{BA}^{VLB}$	The velocity for the $\mathbf{p}_{BA}^{VLB}$ at the particular time	4,5
$R_{BA}^{VRB}$	The radius of rotation of the right end robot trajectory	4,5

$R_{BA}^{VLB}$	The radius of rotation of the left end robot trajectory	4,5
$VOP_{AB}^{\kappa}$	The velocity obstacle region of an agent A with respect to B by the proposed algorithm at time $\kappa$	4,5
$VOP_{AB}$	The velocity obstacle region of an agent A with respect to B by the proposed algorithm over time	4,5
$\mathbf{v}_{LHP}$	The selected velocity outside of $VOP_{AB}$ and within the left half plane of the accessible velocity region	4
$\mathbf{v}_{RHP}$	The selected velocity outside of $VOP_{AB}$ and within the right half plane of the accessible velocity region	4
$\bar{R}_{BA}^{rot}$	The average of $R_{BA}^{rot}(t)$	4
$C_L, C_R$	The calculated evaluation function of $\mathbf{v}_{LHP}$ and $\mathbf{v}_{RHP}$	4
$L$	The distance between two wheels of the two-wheeled robot	4,6
$v_l, v_r$	The left and right speed of each wheel of the robot	4,5
$\mathbf{O}_{VRB}$	The center of robot's right end circular trajectory	5
$\mathbf{O}_{VLB}$	The center of robot's left end circular trajectory	5
$a_{max}$	The maximum magnitude of the acceleration of the robot	6

---

# **Chapter 1**

## **Introduction**

### **1.1 Background and Motivation**

Moving obstacle avoidance in dynamic environments has been studied in robotics for several decades. Typically, navigation algorithm of the robot is divided into two categories: centralized and decentralized navigation systems [1]. In a centralized system, a single control system supervises the entire environment including numerous mobile robots and moving obstacles [2, 3, 4, 5]. Because only a single system handles the environmental information and generates proper path or velocity for each robot, the performance of the centralized system is deeply concerned with the computational complexity. If the number of moving obstacles increases, operating area becomes larger, or robots' missions are diversified, then the computational complexity will

increase exponentially [6, 7]. When increasing the number of robots to be controlled, the obstacle avoidance problem may be NP-complete and NP-hard [8].

Besides, each robot decides its own motion in decentralized navigation system. Because the overall situation is not considered, optimal path cannot be guaranteed. However, the decentralized system is easy to implement and has good scalability and applicability for a variety of systems. In this reason, decentralized navigation system has been extensively and constantly treated. The robot should decide its velocity in accordance with the situation using several rules and criteria, because the robot can only use its local environmental information by installed sensors.

Most recently, a number of research groups are aiming to build a hybrid control system by combining the centralized and decentralized navigation algorithms [9, 10, 11]. However, the current researches do not totally fuse two algorithms. They just perform each algorithm step by step. Thus, enhancing the performance of the decentralized algorithm is still meaningful.

The velocity obstacle approach [12] is a fundamental research of the reactive obstacle avoidance problem in decentralized navigation of a mobile robot. The robot predicts obstacles' velocities and calculates the collision cone where the robot may collide with obstacles in the near future, and then the robot controls its velocity to avoid collision. The velocity obstacle approach has been extended in various forms. Avoidance of an obstacle which



has non-linear velocity is suggested in [13], but the non-linear velocity obstacle technique needs prior knowledge about the future trajectory of the obstacle. Other related concepts are proposed such as generalized velocity obstacles [14] and probabilistic velocity obstacles approaches [15, 16] which consider uncertainty of perception based on probability concept.

Including the above mentioned techniques, most of the velocity obstacle algorithms represent velocity of the robot in the configuration space with Cartesian coordinates. However, environmental information acquired by sensors is represented by polar coordinates whose origin is the center of the robot. The movement of the robot is expressed as linear velocity and angular velocity, so representing robot velocity in the polar coordinate system makes check kinematic and dynamic constraints of the robot convenient. Furthermore, when we design the motion planning processes by adopting polar coordinates, several coordinate conversion steps can be skipped. In this regard, this dissertation focuses on analysis of a motion planning algorithm based on robot-centered polar coordinates.

Several literatures have used polar coordinated velocity vector for navigation. The robot selects its velocity using velocity space presented in [17]. The velocity space is a polar coordinate system which consists of linear and angular velocities, so kinematic and dynamic constraints of the robot can be simply considered. However, obstacles only have linear velocities unlike the robot, and the collision region calculation becomes more complex as the

number of the robot increases. Furthermore, the velocity selection part has quite heuristic points when the robot corrects its velocity from inside of the collision region to outside of the collision region. Another velocity selection algorithm using occupancy grid is recently suggested [18]. According to this algorithm, the robot calculates its velocity using the weights which reflect perception uncertainty in grid space. Consequently the robot may not get convergence results using the algorithm of [18]. In other words, the robot cannot generate the same path for navigation even in the same condition. It means that the proposed algorithm of [18] may generate safe trajectory for the robot, but cannot guarantee fast or minimum time trajectory.

## **1.2 Related Works**

A brief overview of local and reactive navigation technique and velocity obstacle approach is presented in this section. Global path planning is an approach that generates the complete path of the robot to the goal point such as [19] and [20]. The robot plans its optimal path using all the information in the environment, so global path planning is employed for the centralized navigation system only.

Reactive navigation plans the path of the robot to its goal by reacting only to its local environment at every time step. By the decentralized navigation system, each robot calculates its path itself, so reactive navigation is usually employed for the decentralized navigation system. There are classical reactive

navigation algorithms such as vector field histogram [21] and dynamic window approach [22]. The situation cannot avoid collision is defined as Inevitable Collision States (ICS) in [23], and a navigation algorithm to derive ICS-free has researched in [24]. A navigation algorithm focusing on robot safety in route environments has developed in [25], but it limits the situation for static obstacles.

The velocity obstacle approach [12] is a successful and widely used concept for local and reactive navigation, and it is based on the collision cone [26]. The original concept of the collision cone is represented by the range of angles which is a set of straight lines from a point robot to points on a circular object's contour. In [12], the collision cone is mapped in the Cartesian coordinated velocity space and transited by the velocity of the obstacle. Then, the velocity of the robot is determined among the vectors which do not pass the collision cone. The velocity obstacle approach has been employed in practice for a robotic wheelchair [27], a robot within a pharmaceuticals plant [28] and more.

Most of the variations of the velocity obstacles represent the collision cone and the robot velocity in Cartesian coordinated form due to the transition term, though the original collision cone is represented in the angular coordinated values. However, we can get profits in robot motion control by robot-centered polar coordinated form. Generally, the robot has kinematic and dynamic constraints defined from the center of the robot. Kinodynamic

constraints of the robot restrict the range of velocity and acceleration, consequently the traversable region is also limited. Robot motion in the traversable region with the constraints may be difficult to represent formulaic forms in Cartesian coordinates. In this reason, recently, several literatures have tried to do not use the Cartesian coordinate system.

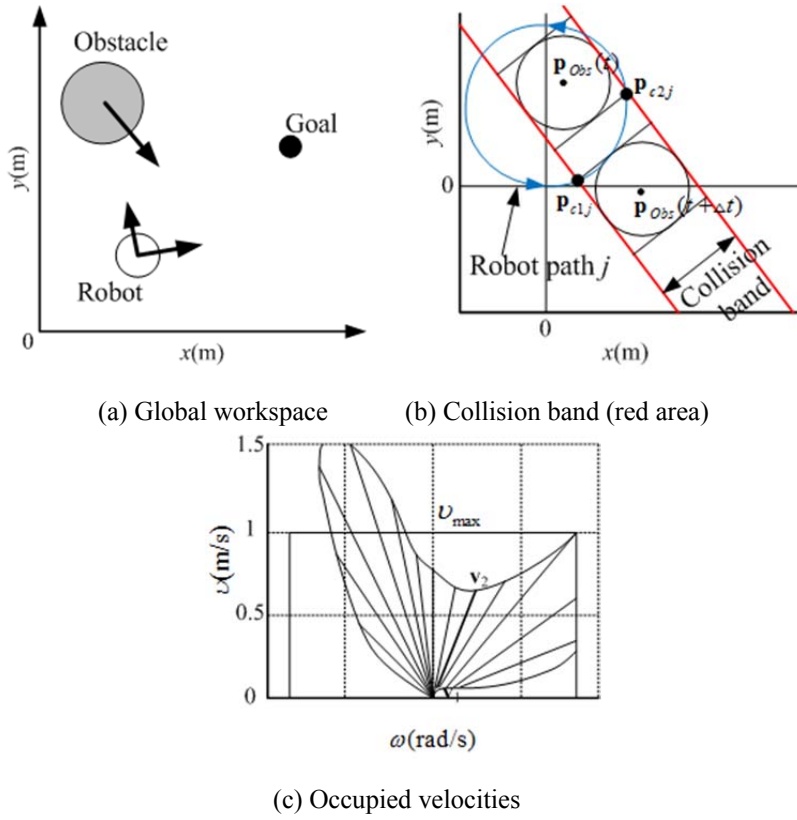


Figure 1.1 Obstacle avoidance scheme of the robotic motion planner using the velocity space [17]: (a) a mobile robot and a moving obstacle in workspace, (b) a collision band in the configuration space and two collision points  $p_{c1j}$  and  $p_{c2j}$  for robot path  $j$ , (c) a dynamic objects velocity set on the plane  $(\omega, v)$ .

The difference of coordinates is a motivation of an obstacle avoidance research called dynamic velocity space [17]. The collision cone used in the original velocity obstacle approach is not applied in the dynamic velocity space algorithm, but a collision band which is similar to the collision cone is developed as shown in Figure 1.1 (b). The collision band is a zone swept by the obstacle that has linear velocity. The collision band is projected in robot's velocity space composed by linear and angular velocities, and then a set of robot's velocities which lead collision with the obstacle is obtained as the shaded region in Figure 1.1 (c). The set of robot velocities is called a dynamic object velocity set. When operating the robot with circular path, the velocity of the robot is represented as a simple straight line which starts from the origin of the velocity space of the robot.

However, there are several limitations of the dynamic velocity space. First of all, the obstacle can move only straight path with a constant velocity. Second, deciding new goal when the velocity of the robot meets the projected region of the collision band is described in heuristic manner and is not explained concretely. Moreover, the algorithm is designed to change the current robot velocity to the new velocity by passing through several phases. However, these intermediate phases of velocity are not significant in the actual operation. The intermediate phase calculated in the previous time step may be useless, because the situation around the robot changes every time

step and the collision region may be changed also. For example, if the direction or number of obstacles changes before the robot reaches the desired velocity, the intermediate velocity phases might lead a conflict, because the dynamic velocity space approach is not designed to guarantee collision-free of the intermediate phases. Therefore, driving the robot to the desired velocity as close as possible within the constraints can be practical in order to take advantages of local and reactive navigation instead of calculating the intermediate phases.

In Table 1.1, the characteristics of several reactive obstacle avoidance algorithms stated in this section are summarized. The second column of the table means whether the method has been validated through real experimental trials or not. Because the listed algorithms are based on the reactive control method, local minima, described in detail on next section, is inherent property of all of the listed algorithms. When the algorithm requires knowledge of the moving obstacles such as positions or velocities from sensors other than a range scanning sensor, the fourth column is filled with ‘required’. The last column represents the existence of constraints on mobility of the obstacles. The vector field histogram method considers only static obstacles, so the fourth and fifth columns do not need.

TABLE 1.1  
PROPERTIES OF REACTIVE OBSTACLE AVOIDANCE ALGORITHMS

Algorithm	Experimental Results	Local Minima	Sensors Other Than Range Sensor	Restriction on Obstacle Velocity
Vector Field Histogram [21]	Yes	Yes	Only Stationary	Obstacles
Dynamic Window [22]	No	Yes	Required	No
Inevitable Collision States [23]	No	Yes	Required	No
Velocity Obstacles [12]	Yes	Yes	Required	No
PVO [14, 15]	No	Yes	Not Required	Yes
Dynamic Velocity Space [17]	No	Yes	Not Required	Yes

### 1.3 Contribution

The main purpose of this dissertation is to develop an algorithm that is leading the robot quickly to the destination and satisfying performance evaluation indices for the robot path in the dynamic environments. In this

dissertation, ‘performance evaluation indices for the robot path’ implies three meanings. The first one is collision-free navigation, the second is smooth path generation, and the last one is local minima avoidance. Considering these factors, this dissertation presents an obstacle avoidance algorithm that represents collision region mathematically on robot-centric polar coordinates and considers continuity of the trajectory the robot.

In order to generate the collision-free path of the robot, the velocity obstacle approach is used here. As briefly introduced above, the velocity obstacles concept guarantees collision-free of the robot. Recently, the velocity obstacle approach is extended for real applications by considering non-linear velocity and sensor noise and adopting a feedback control technique, but their interpretations are based on Cartesian configuration space which is familiar with human users. Actually, most of the robot performs tasks automatically using acquired information based on the local frame. Because the robot, which uses decentralized navigation system, does not communicate with a central controller and performs tasks individually, the robot does not need to convert the motion planning results from the local frame information to the global frame.

Therefore, this dissertation interprets the velocity obstacles concept based on robot-centered polar coordination which is the same as the local frame of each robot. As a result, the robot can process sensor data and represent commands for hardware platform concisely. Furthermore, since the robot



generally has kinematic and dynamic constraints defined from robot's center, reflecting kinodynamic constraints to the traversable region of the robot in formulaic forms using robot-centered polar coordinates is prompter than the case of Cartesian coordinates.

Smoothness of the path can be categorized into two factors: oscillation-free and less sudden changes of velocity. These factors cause frequent changes in velocity, so slip errors and cumulative odometry errors can be increased. Oscillation will occur when the robot, the moving obstacle, and the destination are on a straight line, and the moving directions of the robot and the obstacle are interrupting each other. Sudden change of robot's velocity is usually caused when multiple obstacles are in a tangle or the difference between robot's actual velocity and the calculated velocity by navigation algorithm occurs. If the velocities of the robot and the obstacles are exactly the same as the expected value, situations like sudden velocity change do not occur. In the case that the velocity, which the robot wants to have, overlaps with the collision region, the robot should adjust its velocity. Then the velocity of the robot will change from the primary value. Moreover, by applying various constraints, the relation between the robot and the obstacles of the next time step may differ from the relation of the previous time step.

The proposed algorithm of this dissertation predicts about future conditions using various factors such as shape of collision region from velocity obstacles, the current velocity of the robot, and others. By reflecting a

lot of information on the velocity decision process, oscillation of robot's trajectory and sudden velocity changes can be reduced experimentally than the conventional algorithms. Furthermore, smoothness is examined using geometric continuity [30, 31] of robot trajectories, since, the curve, which is  $G^1$  and  $G^2$  continuous, does not have mechanical oscillations.

One of the worst case of obstacle avoidance is moving back again due to a lack of space to move in front of the robot, and we call it local minima in this dissertation. The robot with the conventional velocity obstacle algorithm decides its velocity vector that corresponds to the local minimum when the robot is surrounded by the collision region in the coordinate system of the robot's velocity space. This situation mainly happens when a new obstacle is detected in the sensing range or the robot fails to reflect obstacles' movement to the velocity decision process correctly. Because velocity obstacle approaches are based on reactive methods, local minima cannot be totally eliminated. However, it can diminish by in-depth prediction about situations of future time steps.

The main purpose of this dissertation is driving the robot to its destination with a high level of achievement in performance evaluation indices without any collision in dynamic environments. To achieve the purpose, this dissertation develops a new reactive obstacle avoidance algorithm which re-analyzes the velocity obstacle approach and represents the collision region mathematically on robot-centered polar coordinates. By predicting changes of

circumstances accurately, the new algorithm prevents frequent robot's velocity changes, so the smoothness and local minima avoidance of path planning are improved.

## **1.4 Organization**

This dissertation is organized as follows. In Chapter 2, background and fundamental formulations of the conventional velocity obstacles approaches are described. Chapter 3 defines the problem of reactive control in polar coordination and states basic assumptions. The developed algorithm of the velocity obstacles concept in polar coordinates is formulated in Chapter 4, and its analysis from the mathematical viewpoint is described in Chapter 5. The performances of the proposed algorithm are validated in dynamic environments with numerous scenarios in Chapter 6. The conclusions of this dissertation are given in Chapter 7.

## **Chapter 2**

# **Velocity Obstacle Approaches**

The development of this thesis is inspired by the long term avoidance of moving obstacles by the velocity obstacles concept [12] and the first attempt to interpret in the linear and angular velocity space of the robot [17]. In this chapter, details of the related studies are provided including formulations and discussions.

### **2.1 Velocity Obstacles**

The concept of the velocity obstacle approach is introduced by Fiorini and Shiller [12]. The velocity obstacle approach is used to select a velocity for the robot A such that collisions with the obstacle B are avoided, assuming that this velocity can be adopted instantaneously.

The current position of a robot A and a moving obstacle B are represented

as  $\mathbf{p}_A^{cart}$  and  $\mathbf{p}_B^{cart}$  in Cartesian coordinates. For simplicity, the robot and the obstacle are modeled as circular objects with radii  $R_A$  and  $R_B$ , respectively. Fiorini and Shiller assumed that the robot and the obstacle move without rotations, and that the obstacle moves along arbitrary trajectories, and that their instantaneous position and velocity are measurable.

The collision region by the velocity obstacles is represented as  $VO_{AB}^\kappa$ , and it is a cone-shaped region.  $VO_{AB}^\kappa$  is a set of all relative velocities of the robot A with respect to the obstacle B that will lead collisions between A and B before time  $\kappa$ .

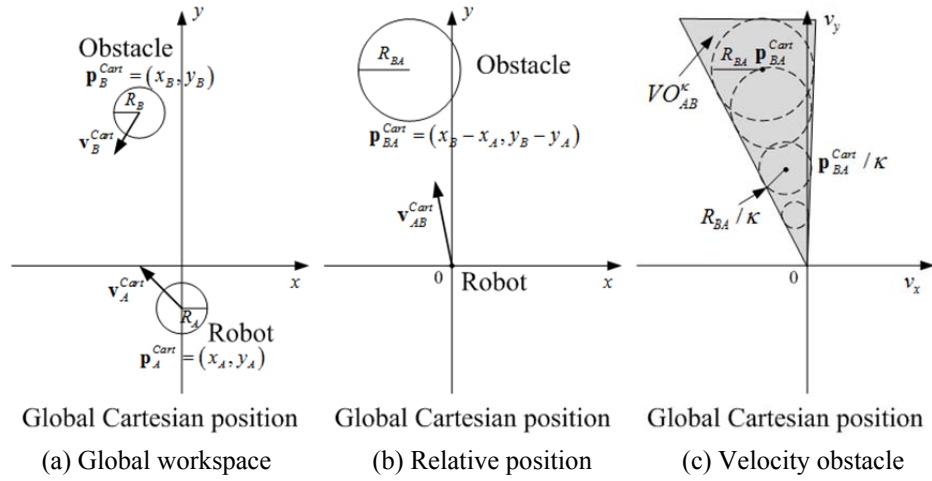


Figure 2.1 A graphical procedure of the original velocity obstacle algorithm: (a) a mobile robot and a moving obstacle in global Cartesian configuration space, (b) the transformation into the coordinates relative to the robot, (c) the velocity obstacle region  $VO_{AB}^\kappa$  as a union of discs.

The relation between the robot and the obstacle in Cartesian configuration space as in Figure 2.1 (a) can be represented in relative coordinates as in Figure 2.1 (b). Then,  $\mathbf{p}_{BA}^{Cart} = \mathbf{p}_B^{Cart} - \mathbf{p}_A^{Cart}$  becomes a current relative position of the obstacle B with respect to the robot and  $R_{BA}$  is defined as a sum of  $R_B$  and  $R_A$ . When the robot has certain relative velocity  $\mathbf{v}_{AB}^{Cart}$  which satisfies following equation, collisions between the robot and the obstacle will occur at time  $t$ .

$$\left\| O + \mathbf{v}_{AB}^{Cart} t - \mathbf{p}_{BA}^{Cart} \right\| < R_{BA} \quad (2.1)$$

The equation is rearranged for non-zero  $t$ :

$$\left\| \mathbf{v}_{AB}^{Cart} - \frac{\mathbf{p}_{BA}^{Cart}}{t} \right\| < \frac{R_{BA}}{t}. \quad (2.2)$$

Equation (2.2) forms discs which have center points at  $\mathbf{p}_{BA}^{Cart}/t$  with radii  $R_{BA}/t$ . When  $\mathbf{v}_{AB}^{Cart}$  satisfies this inequality, it is included in the collision region  $VO_{AB}^\kappa$ , therefore  $VO_{AB}^\kappa$  becomes a union of discs as described in Figure 2.1 (c) and below equation:

$$VO_{AB}^\kappa = \bigcup_{0 < t \leq \kappa} D\left(\frac{\mathbf{p}_{BA}^{Cart}}{t}, \frac{R_{BA}}{t}\right). \quad (2.3)$$

Geometrically,  $VO_{AB}^\kappa$  is interpreted as a cone with its apex at the origin at the origin in relative velocity space. The velocity obstacles approach assumes that the obstacle B is moving at a constant velocity  $\mathbf{v}_B^{Cart}$ . Then,  $VO_{AB}^\kappa$  will be shifted about the velocity of the obstacle, so it becomes  $VO_{AB}^\kappa \oplus \{\mathbf{v}_B^{Cart}\}$  where  $\oplus$  means the Minkowski sum. The definition of the velocity obstacles

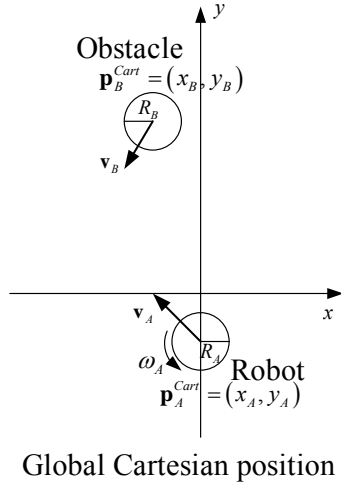
implies that if the robot chooses its velocity  $\mathbf{v}_A^{Cart}$  outside  $VO_{AB}^\kappa \oplus \{\mathbf{v}_B^{Cart}\}$ , the robot is guaranteed not to collide with the obstacle B before time  $\kappa$  if the robot and the obstacle maintain their velocities for at least  $\kappa$ . If the chosen velocity of the robot is not reachable due to acceleration or kinematic constraints, collision avoidance is not guaranteed.

## 2.2 Dynamic Velocity Space

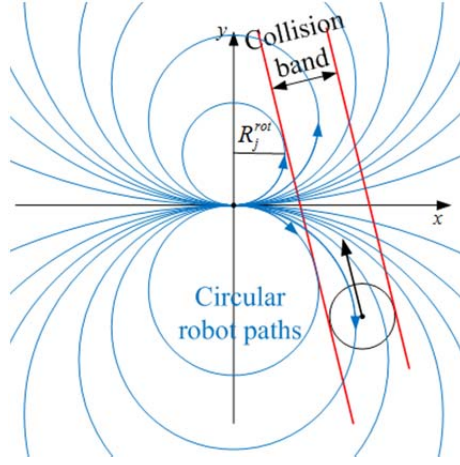
Adding constraints to velocity space based obstacle-avoidance methods, has been considered by Owen and Montano [17]. The dynamic environments and the non-holonomic and dynamic constraints of the robot are represented in the velocity space  $(v, \omega)$  in [17]. Then, the robot can compute its motion commands directly in this space. The proposed algorithm in [17] is named the dynamic velocity space algorithm.

Owen and Montano solve for the time at which a robot with certain velocity and a moving obstacle will arrive at the same location in order to differentiate between safe and collision causing robot angular velocities. The dynamic velocity space method is similar to the original velocity obstacles approach, but there exist several differences. The constraints of the robot and the obstacles are different from the velocity obstacles.

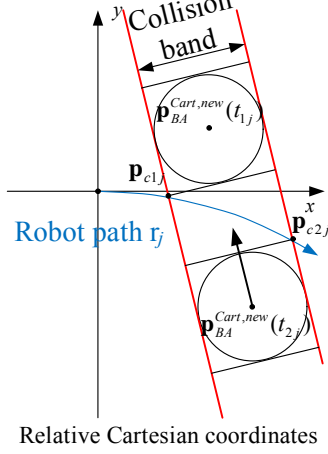
- The robot can move straight or circular paths.
- The obstacle can move straight paths with a constant velocity.
- The moving obstacles are represented as polygons.



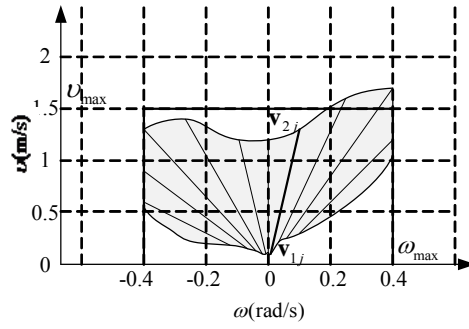
(a) Global work space



(b) Robot paths and collision band



(c) Two intersecting points of  $r_j$



(d) Velocity set leading collisions

Figure 2.2 A graphical procedure of the dynamic velocity space algorithm: (a) a mobile robot and a moving obstacle in global Cartesian configuration space, (b) circular robot paths(blue) intersecting with a collision band(red) in robot-centered Cartesian coordinates, (c) a robot path  $r_j$  and its two intersecting points  $\mathbf{p}_{c1j}$  and  $\mathbf{p}_{c2j}$ , (d) a dynamic object velocity set on the linear and angular velocity plane and sample velocities  $\mathbf{v}_{1j}$  and  $\mathbf{v}_{2j}$  calculated from  $\mathbf{p}_{c1j}$  and  $\mathbf{p}_{c2j}$ .



By assuming that the robot maintains the linear and angular velocities during a sampling period, the first constraint seems reasonable. The sample paths of the robot are described in Figure 2.2 (a). The second constraint is acceptable when the obstacle motion is considered as piecewise straight lines. However, it can be eased, and relaxing the second constraint is discussed in next chapter. The last constraint is a general constraint.

In [17] a collision band, zone swept by the obstacle moving along a straight line, is designed. For a specific path of the robot  $r_j$ , one or two points are intersecting with the boundary of the collision band as presented in Figure 2.2 (b). When the robot passes two intersecting points  $\mathbf{p}_{c1j}$  and  $\mathbf{p}_{c2j}$ , the corresponding time is  $t_{1j}$  and  $t_{2j}$ . If the obstacle passes  $\mathbf{p}_{c1j}$  at time  $t_{1j}$  and  $\mathbf{p}_{c2j}$  at time  $t_{2j}$  as in Figure 2.2 (c), this pair of points and their associated times are registered as a collision set. For circular path  $r_j$ , the radius of rotation  $R_j^{rot}$  is fixed, and then the angular and linear velocities are determined as follows:

$$\omega_{ij} = \theta_{ij} / t_{ij} \quad (2.4)$$

$$\nu_{ij} = R_j^{rot} \omega_{ij} \quad (2.5)$$

where  $\theta_{ij}$  is a robot angular displacement on  $R_j^{rot}$  to reach  $\mathbf{p}_{cij}$ .

The robot velocities  $\mathbf{v}_{1j}(v_{1j}, \omega_{1j})$  and  $\mathbf{v}_{2j}(v_{2j}, \omega_{2j})$  and their corresponding times  $t_{1j}$  and  $t_{2j}$  are mapped in the velocity space of the robot as presented in Figure 2.2 (d). If the robot has the linear and angular velocities

along the straight line between  $\mathbf{v}_{1j}$  and  $\mathbf{v}_{2j}$ , the robot collides with the obstacle. These calculations are extended to the whole space for all  $j$ , and then the set of straight lines forms Dynamic Object Velocity (DOV). The union of all the zones of DOV for all the obstacles provides the DOV set and represents the velocities for which could have collision if they were maintained for some time.

The lower limits of DOV set, for example  $\mathbf{v}_{1j}$  in Figure 2.2 (d), involve the maxima robot velocities to allow the obstacle pass before the robot, and the upper limits, for example  $\mathbf{v}_{2j}$  in Figure 2.2 (d), represent the minima robot velocities to escape before the obstacle pass. As a consequence, selecting a velocity outside DOV set implies that there is no collision between the robot and the obstacles during the whole time horizon.

Using the angle between the current robot orientation and the direction of the goal in the robot-centered frame, the angular velocity to the goal can be obtained. Under the maximum linear velocity  $v_{max}$  and maximum angular velocity  $\omega_{max}$ , the velocity towards the goal  $\mathbf{v}_{des}$  is determined. The dynamic velocity space algorithm always tries to have the maximum linear velocity value, so generally  $\mathbf{v}_{des}$  is  $(v_{max}, \omega)$  as in Figure 2.3.

If  $\mathbf{v}_{des}$  does not intersect with the DOV set, the robot will move with  $\mathbf{v}_{des}$ . Otherwise, the robot chooses the nearest velocity to  $\mathbf{v}_{des}$  or  $\mathbf{v}_{SP}$ , and it is represented as  $\mathbf{v}_{MT}$ . A velocity  $\mathbf{v}_{SP}$  has the same angular velocity as  $\mathbf{v}_{des}$  and the largest linear velocity among the lower limits of DOV set.



dynamic velocity space algorithm does not insure that  $\mathbf{v}_d$  does not influence collisions.

## Chapter 3

# Problem Description of Reactive Control

### 3.1 Reactive Control in Cartesian and Polar Coordinates

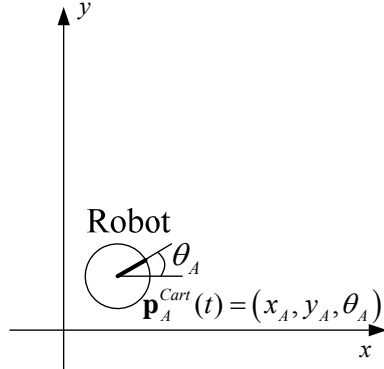
The difference between global Cartesian coordinates and robot-centered polar coordinates of reactive control is simply described in this section. Generally, reactive navigation control of the mobile robot consists of a few steps such as environmental change perception, collision prediction, motion planning, and robot control. When motion planning part is implemented in Cartesian coordinates, reactive navigation follows below schemes. The procedure is graphically represented in Figure 3.1. The positions and velocities of the robot and obstacle in Cartesian coordinates are extended by including the orientation angle, for example  $\mathbf{p}_A^{cart}$  is represented as  $(x_A, y_A, \theta_A)$ .

- Step A-1) Robot localization in global Cartesian coordinates

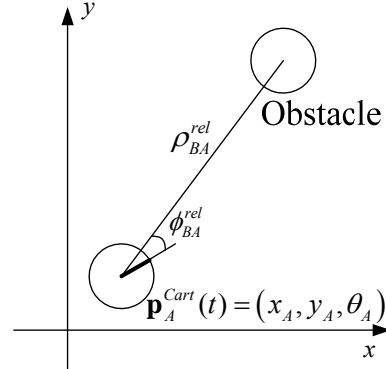
- Step A-2) Obstacle sensing in robot-centered polar coordinates
- Step A-3) Conversion of the obstacle information into global Cartesian coordinates
- Step A-4) Obstacle pose and velocity estimation in global Cartesian coordinates
- Step A-5) Robot velocity decision in Cartesian representation
- Step A-6) Conversion of the new velocity into polar coordinates of the robot
- Step A-7) Transmission of the new velocity to the robot

First, robot localization is performed in global Cartesian coordinates using robot odometer or other external sensors (Step A-1). After that, sensor data represented in robot-centered coordinates should be mapped into global coordinates (Step A-2 through A-4). In local and reactive navigation case, the robot is controlled using information of sensors attached to the robot rather than information received from an external system. Therefore, Step A-3 is necessary. After performing collision prediction and motion planning in the same coordinate system, the robot velocity which can prevent collision is obtained (Step A-5). Robot driving part is operated based on robot-centered frame, so users generally control the robot using translational velocity and rotational velocity. Thus, the robot velocity is converted from the Cartesian

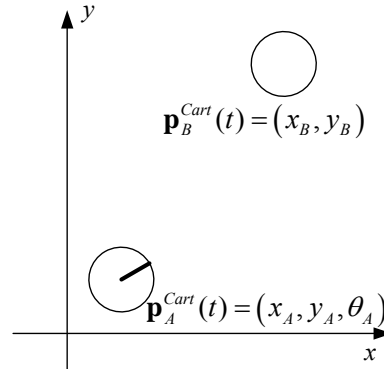
coordinates to robot-centered polar coordinates (Step A-6), and then transmitted to the robot as a control command (Step A-7).



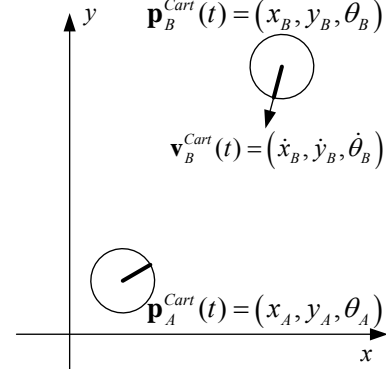
(a) Localization



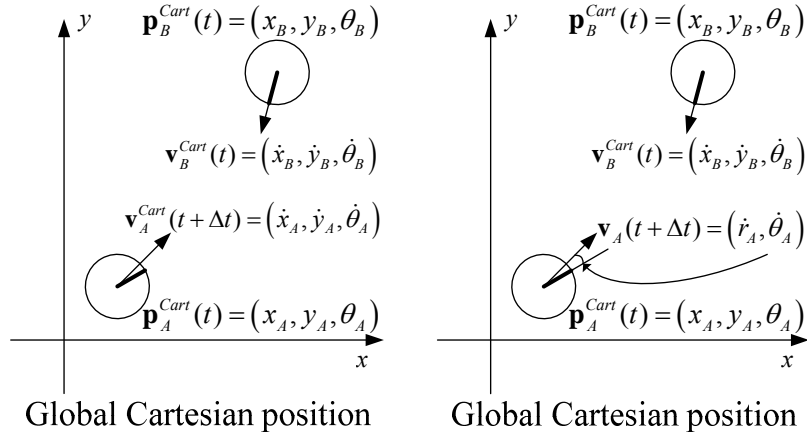
(b) Sensing obstacle



(c) Obstacle position calculation

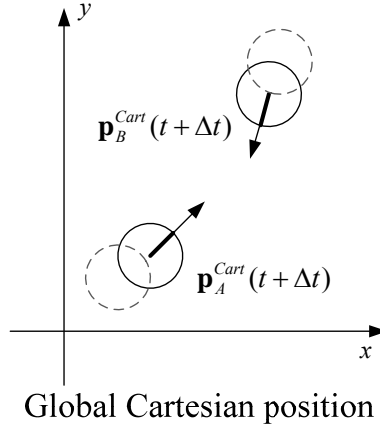


(d) Obstacle velocity calculation



(e) Robot velocity decision

(f) Velocity transformation



(g) New positions at the next time step

Figure 3.1 Motion planning process in global Cartesian coordinates: (a) robot localization, (b) obstacle sensing, (c) conversion of obstacle position into Cartesian coordinates, (d) obstacle pose and velocity estimation, (e) robot velocity decision, (f) conversion of the robot velocity into polar coordinates, (g) new positions at time  $t + \Delta t$ .



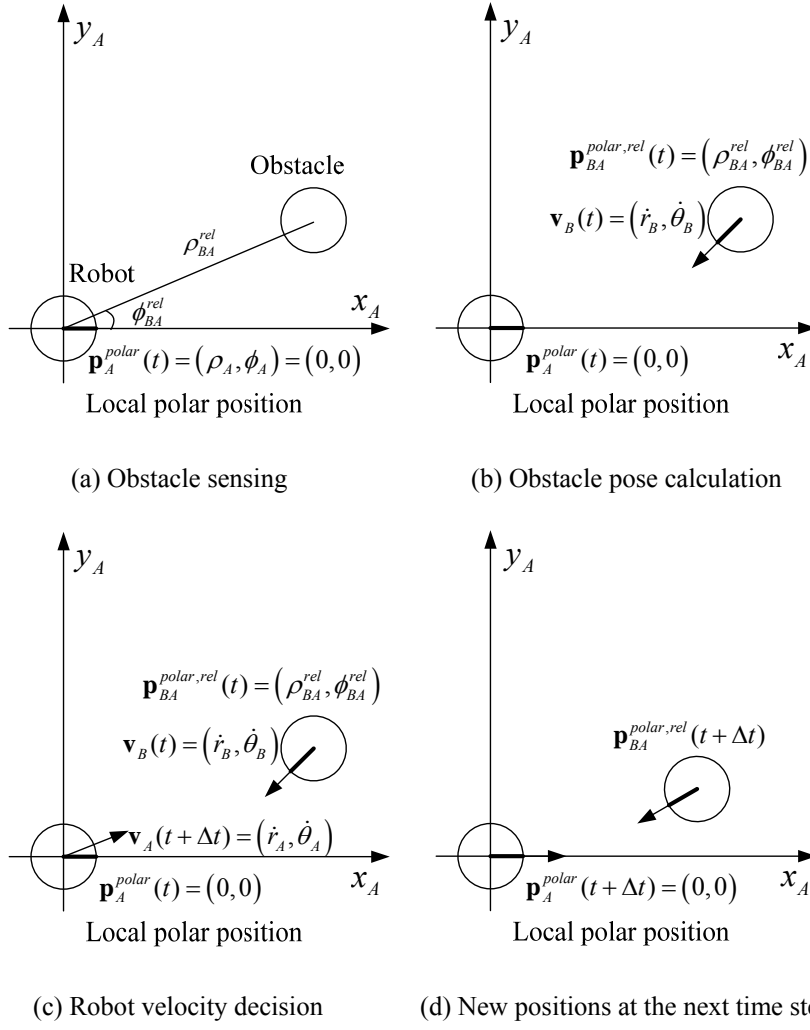


Figure 3.2 Motion planning process in robot-centered polar coordinates: (a) obstacle sensing, (b) estimation of obstacle pose and velocity, (c) robot velocity decision, (d) the relative position at time  $t+\Delta t$  after velocity transmission.

On the other hand, if motion planning part is interpreted in polar coordinates, unnecessary process can be eliminated as shown in Figure 3.2

and following steps.

- Step B-1) Obstacle sensing in robot-centered polar coordinates
- Step B-2) Obstacle pose and velocity estimation in robot-centered polar coordinates
- Step B-3) Decision of robot's new velocity in polar coordinated representation
- Step B-4) Transmission of the new velocity to the robot

If the origin of polar coordinates is the center of the robot, robot localization is not required. Since the origin of the coordinates moves along the robot. However, the translation of coordinates should be checked and updated at every time step. Relative position is updated sequentially. Conversion processes like Step A-3 and A-6 are not necessary, because sensor data and new velocity, obtained as a result of motion planning, are represented with respect to the center of the robot. For safe and efficient obstacle avoidance, a motion planning algorithm based on robot-centered polar coordinates is developed in this thesis, and the details will be described in Chapter 4 and 5.

### **3.2 Basic Assumptions and Definitions**

The main purpose of this dissertation is driving the robot safely and efficiently to its goal without any collision in dynamic environments. To

achieve the purpose, this dissertation is aiming obstacle avoidance with keeping high levels of performance evaluation indices of the robot path. As mentioned in Chapter 1.3, ‘performance evaluation indices of the robot path’ implies three meanings: collision-free navigation, smooth path generation, and local minima avoidance. Several following assumptions and definitions for this purpose are stated below. The robot-centered polar coordinate system is applied to the velocity obstacles technique here. The robot and the obstacles are considered as a circular non-holonomic moving object. It means that the robot or obstacle moves along its heading direction. For computational convenience, the robot model is transformed from a circular robot to a point robot, and instead the radius of the obstacle is extended about the radius of the robot. Then, the circular non-holonomic robot is redefined as a point robot with orientation angle. These robot and obstacle models maintain their linear and angular velocities during the time interval  $\Delta t$ , and choose a new velocity at each time step.

The robot A has constant radius  $R_A$ , and its current position at time  $t$  in polar coordinates is  $\mathbf{p}_A^{polar}(t)$ . Current velocity of the robot is  $\mathbf{v}_A(t)$ .  $\mathbf{p}_{goalA}^{polar,rel}(t)$  is the position of the goal relative to the robot position in the robot-centered polar coordinates system, and it is updated at every time step because the robot moves continuously. Robot velocity to move the robot towards the destination is  $\mathbf{v}_{des}(t)$ , and it is also updated at every time as  $\mathbf{p}_{goalA}^{polar,rel}(t)$  changes. The obstacle B has current position  $\mathbf{p}_{BA}^{polar,rel}(t)$  in

robot-centered polar coordinates, and current velocity  $\mathbf{v}_B(t)$ . Both terms are represented in robot-centered polar coordinates, so the values are relative to the robot.  $\mathbf{v}_B(t)$  is calculated by the current and previous obstacle's position. By using  $\mathbf{v}_A(t - \Delta t)$  and the difference of  $\mathbf{p}_{BA}^{polar,rel}(t)$  and  $\mathbf{p}_{BA}^{polar,rel}(t - \Delta t)$ ,  $\mathbf{v}_B(t)$  can be obtained. Radius of the obstacle  $R_B$  is redefined as  $R_{BA}$  increased by  $R_A$  from  $R_B$ , and the maximum sensing range that the robot can perceive is set to  $r_{range}$ .

In this dissertation, it is assumed that the robot and the obstacles have constant linear velocity and angular velocity from current time to next time step. In other words, piecewise linearity of the robot and obstacle motion is established. Various kinds of paths such as clothoid and anti-clothoid can be generated by controlling acceleration of the robot, but the generated paths are highly dependent on robot model's constraints and surrounding situations. Thus, piecewise constant velocity is applied here in order to explain the obstacle avoidance concept that can be applied generally.

When changing acceleration of the robot frequently, sensor error may increase due to drastic curve of the robot path. On the other hand, when linear and angular velocities are constant, then the radius of rotation is also constant. The constant radius of rotation may reduce unexpected sensing errors. In this reason, the obstacle avoidance motion of this dissertation basically generates circular paths to the robot. Additionally, it is only considered that the robot finds the obstacles when the robot has a non-zero linear velocity. Generally,

the velocity obstacle approaches cope with the situation during driving well. If the stopped robot perceives the obstacle, the robot has many alternatives for obstacle avoidance: turning, waiting, reversing, and so on.

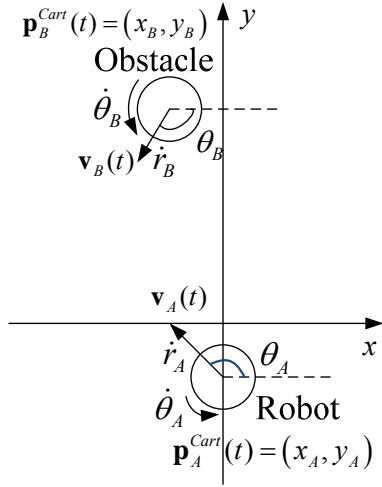
## Chapter 4

# Velocity Obstacles in Polar Coordinates

This section presents a new velocity obstacles approach by interpreting the velocity obstacles concept in the robot-centered polar coordinated system.

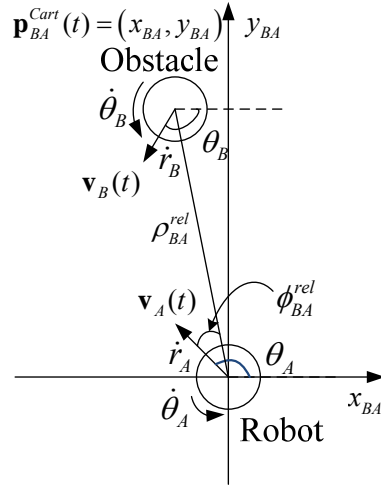
### 4.1 Robot-centered Polar Coordinate Representation of an Obstacle

In Figure 4.1 (a), a situation that the robot A with velocity  $\mathbf{v}_A(t)$  and the obstacle B with velocity  $\mathbf{v}_B(t)$  are moving together is represented in a Cartesian coordinate system. A robot-centered polar coordinate system, the same as a frame that the robot perceives the environments, is represented as a coordinate system whose horizontal axis is robot's heading direction. The situation is described in robot-centered polar coordinates as Figure 4.1 (c).



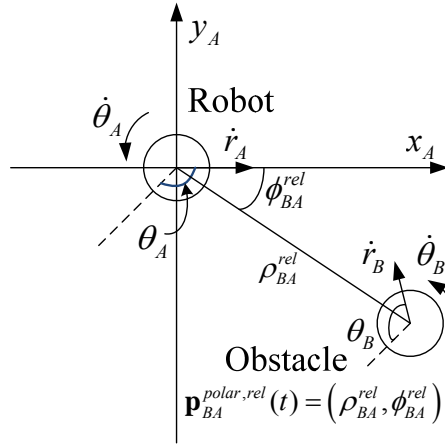
Global Cartesian position

(a) Global workspace



Robot-centered relative Cartesian position

(b) Robot-centered Cartesian representation



Robot-centered polar coordinates

(c) Robot-centered polar representation

Figure 4.1 Representation of the moving robot and obstacle: (a) representation in global Cartesian coordinates, (b) representation in robot-centered relative Cartesian coordinates, (c) representation in robot-centered polar coordinates.

Conventional velocity obstacle approaches have analyzed the collision avoidance problem with relative position in Cartesian coordinates as in Figure 4.1 (b). Cartesian representation is good for users to understand the collision avoidance concept, but it is not promptly available for handling the information expressed from the robot center. Furthermore, robot motion including non-linear velocity is formulated in complex form with Cartesian representation. The robot-centered polar coordinate system is obtained by rotating the coordinates of Figure 4.1 (b) as much as the robot's orientation angle. Since the origin of the coordinates is the robot center, robot constraints and environmental information are applied with the same form that the robot uses.

If an obstacle is detected within the robot's sensing range, current relative position of the obstacle  $\mathbf{p}_{BA}^{polar,rel}(t)$  is represented as  $(\rho_{BA}^{rel}(t), \phi_{BA}^{rel}(t))$ , and current velocity of the obstacle  $\mathbf{v}_B(t)$  is represented as  $(\dot{r}_B(t), \dot{\theta}_B(t))$ . The obstacle position value is exactly the same as relative distance and angle between the robot and the obstacle. If any obstacle is not found within the sensing range, this process is skipped.

## 4.2 Desired Velocity Generation

As mentioned in Chapter 3.2, the robot has piecewise constant linear and angular velocities. Accordingly, the robot's trajectory from the current



position to the goal position becomes parts of circle starting from the current orientation of the robot as shown in Figure 4.2. The center of rotation is always on the vertical axis which represents  $\pi/2$  radian in polar coordinates. The radius of rotation  $R_{goalA}^{rot}(t)$  is maintained during the time interval, so the radius of rotation  $R_{goalA}^{rot}(t)$  which leads the robot to the goal position is obtained by applying the law of cosines to  $\Delta \mathbf{p}_A^{polar} \mathbf{O}_{rot} \mathbf{p}_{goalA}^{polar,rel}$  as follows:

$$\left(R_{goalA}^{rot}(t)\right)^2 = \left(R_{goalA}^{rot}(t)\right)^2 + \left(\rho_{goalA}^{rel}(t)\right)^2 - 2R_{goalA}^{rot}(t)\rho_{goalA}^{rel}(t)\sin\left(\phi_{goalA}^{rel}(t)\right) \quad (4.1)$$

$$R_{goalA}^{rot}(t) = \frac{\rho_{goalA}^{rel}(t)}{2\sin\left(\phi_{goalA}^{rel}(t)\right)}. \quad (4.2)$$

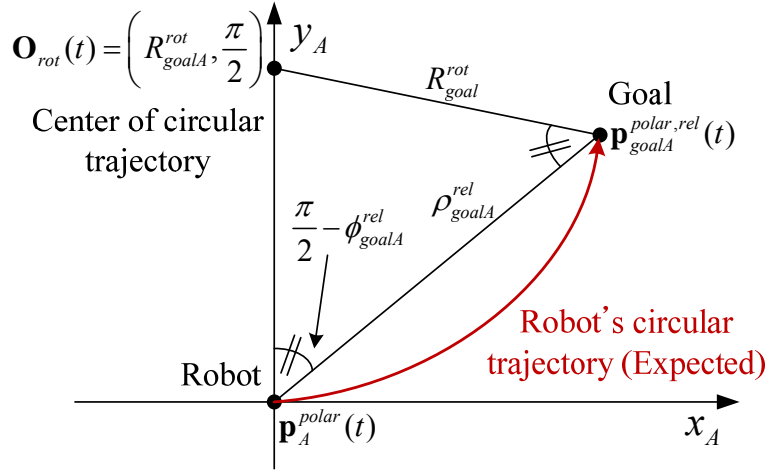


Figure 4.2 The trajectory of the robot from the current position to the goal position. The robot moves along a circular path, because it has piecewise constant linear and angular velocities. The radius of rotation  $R_{goalA}^{rot}$  is calculated using law of cosines.

The robot should maintain the maximum value of linear velocity to reach the destination as fast as possible. Then, angular velocity of the robot  $\dot{\theta}_A(t)$  is the value of linear velocity divided by the radius of rotation, since the point robot A has uniform circular motion over the time interval. Velocity of the robot which guides the robot to reach the destination is represented as in equation (4.3).

$$\mathbf{v}_{des}(t) = (\dot{r}_A(t), \dot{\theta}_A(t)) = (\dot{r}_{A,max}(t), \dot{r}_{A,max}(t) / R_{goalA}^{rot}(t)) \quad (4.3)$$

If there is no disturbance by obstacles and the robot can have the maximum linear velocity immediately, the robot arrives at the goal point with a smooth circular trajectory.

### 4.3 Velocity Obstacle Region Generation

The robot should predict the collision region and react to it after detecting the obstacles. The proposed algorithm generates a velocity obstacle region as a set of all robot velocities that will result in collisions between the robot and the obstacle from the current time  $t$  to a particular time  $\kappa$ . When the robot moves from time  $t$  to  $\kappa$ , the position of the robot  $\mathbf{p}_A^{polar}(\kappa)$  is represented as equation (4.4) with polar coordinates whose origin point is the same as  $\mathbf{p}_A^{polar}(t)$ . Because every element of  $\mathbf{p}_A^{polar}(t)$  has zero value, the robot position is simplified as the latter equation of equation (4.4).

$$\mathbf{p}_A^{polar}(\kappa) = \int_t^\kappa \mathbf{v}_A(\tau) d\tau + \mathbf{p}_A^{polar}(t) = \int_t^\kappa \mathbf{v}_A(\tau) d\tau \quad (4.4)$$

When the distance between the robot position  $\mathbf{p}_A^{polar}(\kappa)$  and the obstacle position  $\mathbf{p}_{BA}^{polar,rel}(t)$  is smaller than  $R_{BA}$ , a collision is occurred. The situation of the robot arriving at the center of the obstacle at a particular time  $\kappa$  can be described in Figure 4.3. In this situation, the robot generates a radius of rotation  $R_{BA}^{rot}(\kappa)$  which leads the robot to the center of the obstacle at time  $\kappa$  as shown in equation (4.2):

$$R_{BA}^{rot}(\kappa) = \frac{\rho_{BA}^{rel}(\kappa)}{2 \sin(\phi_{BA}^{rel}(\kappa))}. \quad (4.5)$$

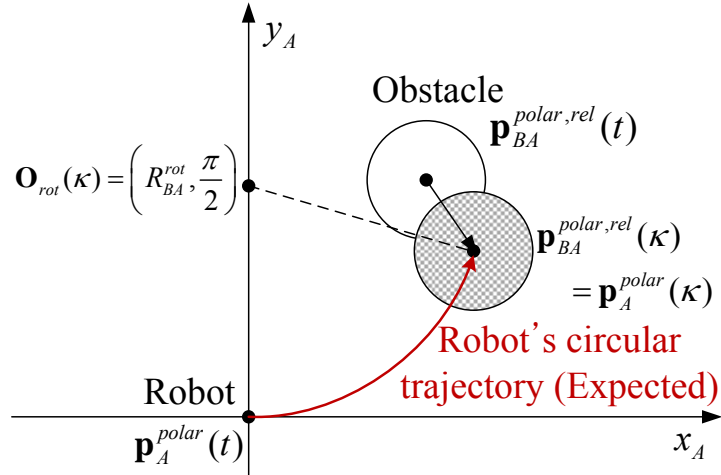


Figure 4.3 The situation of that the robot arrives at the center of the obstacle at time  $\kappa$ . The robot has a radius of rotation  $R_{BA}^{rot}(\kappa)$  and travels along the circular trajectory (red arc) by maintaining the radius of rotation, since the robot follows piecewise linearity.

Since the robot maintains constant acceleration and the radius of rotation, equation (4.6) can be established. As a result, equation (4.5) is rewritten as equation (4.7), and linear velocity of the robot  $\dot{r}_{BA}^{VO}(\kappa)$  is represented as equation (4.8).

$$\dot{\theta}_{BA}^{VO}(\kappa) = \frac{2 \cdot \phi_{BA}^{rel}(\kappa)}{\kappa} \quad (4.6)$$

$$R_{BA}^{rot}(\kappa) = \frac{\dot{r}_{BA}^{VO}(\kappa)}{2 \sin(\dot{\theta}_{BA}^{VO}(\kappa) / 2)} \quad (4.7)$$

$$\dot{r}_{BA}^{VO}(\kappa) = 2 \cdot R_{BA}^{rot}(\kappa) \cdot \sin\left(\frac{\dot{\theta}_{BA}^{VO}(\kappa)}{2}\right) = \frac{\rho_{BA}^{rel}(\kappa) \cdot \sin(\dot{\theta}_{BA}^{VO}(\kappa) / 2)}{\sin(\phi_{BA}^{rel}(\kappa))} \quad (4.8)$$

Therefore, velocity of the robot,  $\mathbf{v}_{BA}^{VO}(\kappa)$ , which makes collisions between the robot A and the obstacle B at time  $\kappa$  becomes as follows:

$$\mathbf{v}_{BA}^{VO}(\kappa) = \left( \dot{r}_{BA}^{VO}(\kappa), \dot{\theta}_{BA}^{VO}(\kappa) \right). \quad (4.9)$$

Velocities which guide the robot to pass the boundary of the obstacle of Figure 4.3 can be obtained in a similar manner. The trajectory of the robot which passes the obstacle boundary is a circle that crosses the robot position (the origin of the coordinates) and the boundary of the robot. The both end trajectories of the robot which passes the outer boundary of the obstacle are parts of two circles that are circumscribed and inscribed on the boundary of the obstacle and have their center position along the vertical axis as shown in Figure 4.4. The end trajectories always start from the origin of the coordinates.

In Figure 4.4, the contact point of the incircle,  $\mathbf{p}_{BA}^{VRB}(\kappa)$ , is on the right side of the obstacle and the contact point of the circumcircle,  $\mathbf{p}_{BA}^{VLB}(\kappa)$ , is on the left side, however the side of the contact points are changing in accordance with positions of the robot and the obstacle.

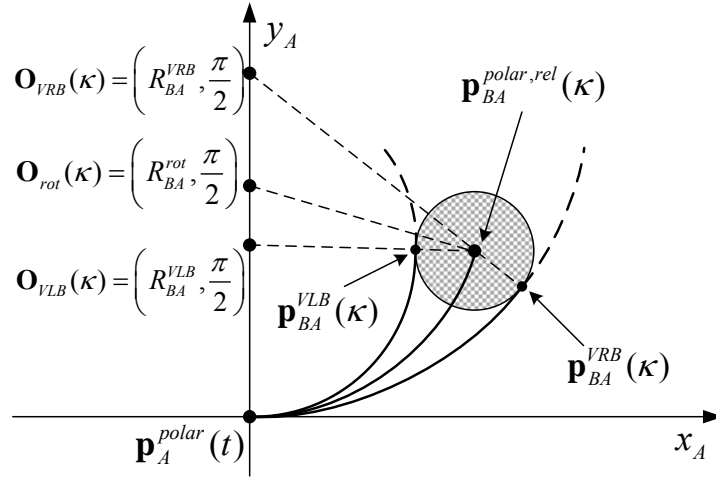


Figure 4.4 The two end boundaries of the robot trajectory which pass the outer boundary of the obstacle. The inscribed point  $\mathbf{p}_{BA}^{VLB}(\kappa)$  becomes a contact point of the robot trajectory and the obstacle boundary. The circumscribed point  $\mathbf{p}_{BA}^{VRB}(\kappa)$  becomes a contact point in the opposite side. Robot velocities, which lead the robot to  $\mathbf{p}_{BA}^{VLB}(\kappa)$  and  $\mathbf{p}_{BA}^{VRB}(\kappa)$ , are the left and right boundary of a velocity obstacle region.

Velocity to pass the left side of the obstacle boundary at time  $\kappa$ ,  $\mathbf{v}_{BA}^{VLB}(\kappa)$ , is calculated as equation (4.10) by using the characteristics of inscription and circumscription and piecewise linearity of the robot motion. Velocity for the right side of the obstacle boundary,  $\mathbf{v}_{BA}^{VRB}(\kappa)$ , can be acquired in the same

manner, and it is represented as equation (4.11). The details of the formula induction including the calculation of  $R_{BA}^{VLB}(\kappa)$  and  $R_{BA}^{VRB}(\kappa)$  are introduced in Section 5.1.

$$\begin{aligned}\mathbf{v}_{BA}^{VLB}(\kappa) &= (\dot{r}_{BA}^{VLB}(\kappa), \dot{\theta}_{BA}^{VLB}(\kappa)) \\ &= \left( 2 \cdot R_{BA}^{VLB}(\kappa) \cdot \sin\left(\frac{\phi_{BA}^{VLB}(\kappa)}{\kappa}\right), \frac{2 \cdot \phi_{BA}^{VLB}(\kappa)}{\kappa} \right)\end{aligned}\quad (4.10)$$

$$\begin{aligned}\mathbf{v}_{BA}^{VRB}(\kappa) &= (\dot{r}_{BA}^{VRB}(\kappa), \dot{\theta}_{BA}^{VRB}(\kappa)) \\ &= \left( 2 \cdot R_{BA}^{VRB}(\kappa) \cdot \sin\left(\frac{\phi_{BA}^{VRB}(\kappa)}{\kappa}\right), \frac{2 \cdot \phi_{BA}^{VRB}(\kappa)}{\kappa} \right)\end{aligned}\quad (4.11)$$

When the robot has its velocity between the two boundary value,  $\mathbf{v}_{BA}^{VLB}(\kappa)$  and  $\mathbf{v}_{BA}^{VRB}(\kappa)$ , the robot collides with the obstacle about  $\kappa$  time later. By varying time  $\kappa$  of the above equations, velocities which make the robot pass the both ends of the obstacle boundary can be represented as two simple curves on the robot-centered polar coordinates of velocity as shown in Figure 4.5. When robot velocity is in the region between the two boundary curves, the robot collides with the obstacle in the near future. The region is called a velocity obstacle region. The velocity obstacle region at time step can be described as equation (4.12), and the velocity obstacle region of the obstacle B over time is defined as equation (4.13).

$$VOP_{AB}^{\kappa} \subset \left[ \int_t^{\kappa} \mathbf{v}_{BA}^{VRB}(\tau) d\tau, \int_t^{\kappa} \mathbf{v}_{BA}^{VLB}(\tau) d\tau \right] \quad (4.12)$$

$$VOP_{AB} = \bigcup_{t < \tau \leq \kappa} VOP_{AB}^{\tau} \quad (4.13)$$

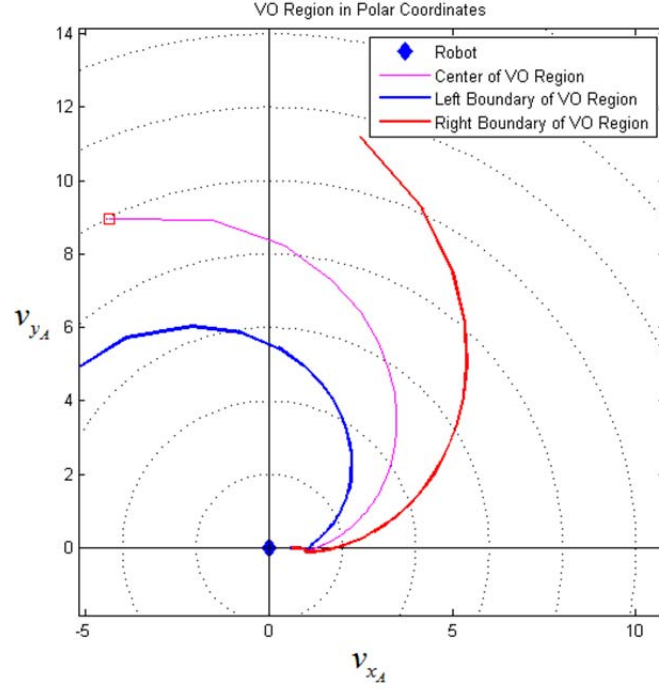


Figure 4.5 Robot velocities toward the left and right boundaries of the obstacle on polar coordinates. The region between two boundaries becomes the velocity obstacle region. Robot velocities of the left boundary (blue curve) is obtained from  $\mathbf{v}_{BA}^{VLB}(\kappa)$  by varying  $\kappa$ . robot velocities of the right boundary (red curve) if obtained from  $\mathbf{v}_{BA}^{VRB}(\kappa)$  by varying  $\kappa$ .

In equation (4.10) and (4.11), the mobility of the obstacles is already reflected by varying the radius of rotation. It means that the velocity of the obstacle is automatically applied in the generation process of the velocity obstacle region. In contrast, other conventional algorithms usually generate

the velocity obstacle region for the stopped obstacle, and then translate it as the velocity of the obstacle. When the robot chooses and maintains its velocity outside of the velocity obstacle region  $\{VOP_{AB}\}$ , the robot A can travel to the destination until time  $\kappa$  without collisions with the obstacle B.

## 4.4 New Velocity Decision

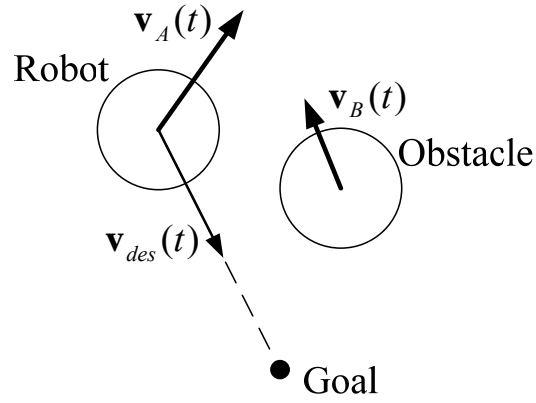
### 4.4.1 Basic Method of Choosing Alternative Velocity

In the case of the situation that the desired velocity of the robot  $\mathbf{v}_{des}(t)$  generated by Section 4.2 does not intersect with the velocity obstacle region generated by Section 4.3 from time step  $t$  to  $\kappa$  (Equation (4.14)), robot velocity at the next time step  $\mathbf{v}_A(t + \Delta t)$  is decided as  $\mathbf{v}_{des}(t)$ .

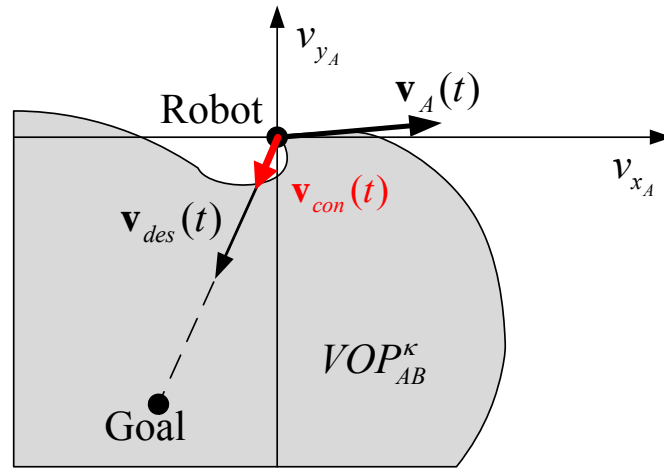
$$\mathbf{v}_{des}(t) \notin VOP_{AB} \quad (4.14)$$

However, if intersection occurs, then the robot should choose its velocity outside of the velocity obstacle region  $\{VOP_{AB}\}$ . In conventional velocity obstacle researches, the robot's new velocity is determined by finding a vector which has the smallest difference from  $\mathbf{v}_{des}(t)$  among vectors outside of the velocity obstacle region. However, the conventional velocity determination method cannot be directly applied to the proposed algorithm of this dissertation.



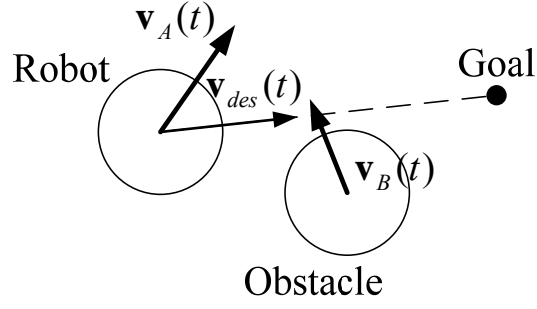


(a) Hurried turning situation

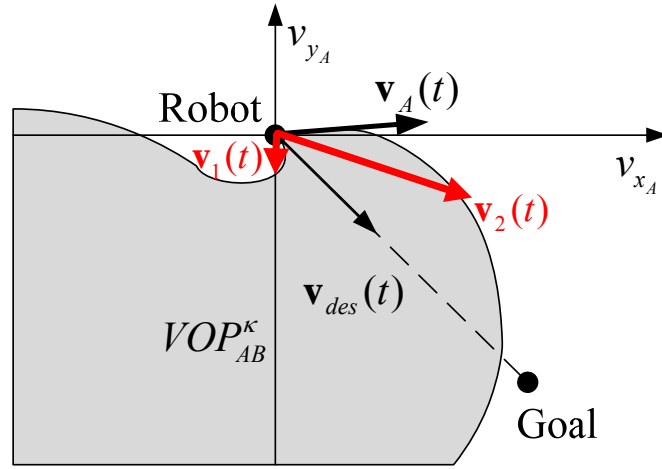


(b) Velocity obstacle regions

Figure 4.6 An example of drastic turning to avoid urgent collisions. In this situation,  $\mathbf{v}_{des}(t)$  is intersecting with  $\mathbf{VOP}_{AB}^k$ , and the nearest velocity vector from  $\mathbf{v}_{des}(t)$  among the region unoccupied by  $\mathbf{VOP}_{AB}^k$  is beyond the constraints: (a) a situation of hurried turning, (b)  $\mathbf{VOP}_{AB}^k$  of the proposed algorithm on the same situation.



(a) Situation of oscillated motion



(b) Velocity obstacle regions

Figure 4.7 An example of oscillation. In this situation,  $\mathbf{v}_{des}(t)$  is intersecting with  $\mathbf{VOP}_{AB}^k$ , and the nearest velocity vector from  $\mathbf{v}_{des}(t)$  among the region unoccupied by  $\mathbf{VOP}_{AB}^k$  is beyond the constraints: (a) configuration of the robot and the obstacle, (b) the two nearest velocities for each side of  $\mathbf{VOP}_{AB}^k$  of the proposed algorithm on the same situation.

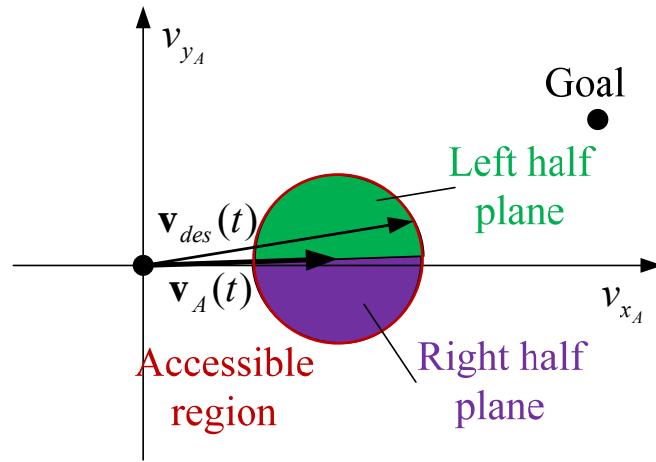
Occasionally, the heading direction of the robot can be almost opposite the desired velocity angle as shown in Figure 4.6, since the robot drastically turns

its direction to avoid urgent collisions. The conventional velocity determination methods do not consider kinodynamic characteristics of the robot, so it decides new velocity of the robot as a red vector in Figure 4.6 (b). The new velocity vector may require impractical changes of velocity to the robot. Then the robot may not close to the value of  $\mathbf{v}_{des}(t)$  in a short time and inevitably collide with the obstacle, or the robot may have zigzag motion due to repeated heading direction switching.

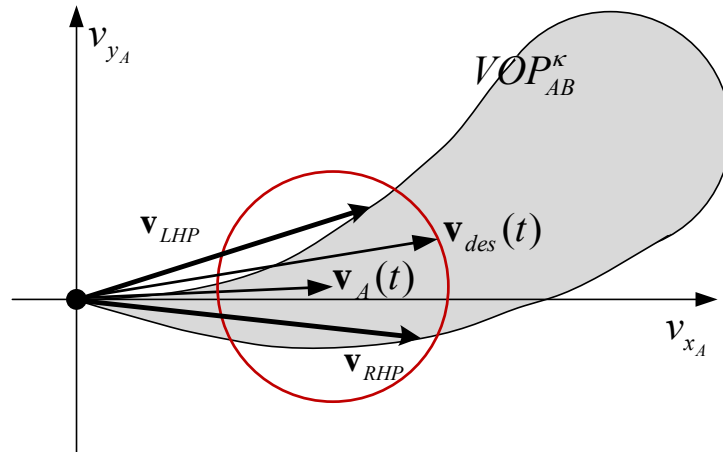
When the desired velocity  $\mathbf{v}_{des}(t)$  is placed in the middle of the velocity obstacle region as shown in Figure 4.7, a vector which is far from the current direction,  $\mathbf{v}_1(t)$  in Figure 4.7 (b), can be the nearest vector from the desired velocity. If the robot selects this vector without any further consideration, it may be confuse which direction is safer. Then the robot oscillates its direction forever. In this case, selecting  $\mathbf{v}_2(t)$  can be a better solution even though the robot travels along the far direction.

#### **4.4.2 Proposed Method: Evaluating Two Alternatives**

In this section, new velocity decision method that considers both components to move the robot toward the goal position and restrain the sudden changes of direction is proposed. As shown in Figure 4.8, the accessible region of robot velocity is a circle which has the radius as the maximum value of acceleration and the center position at the end point of the robot velocity vector  $\mathbf{v}_A(t)$ .



(a) Current velocity, accessible region, and half planes



(b) Selection of  $\mathbf{v}_{LHP}$  and  $\mathbf{v}_{RHP}$

Figure 4.8 The concept of selecting two alternative velocities  $\mathbf{v}_{LHP}$  and  $\mathbf{v}_{RHP}$ : (a) a description of the left half plane and the right half plane, (b) selecting two alternative velocities from the left and right half planes.

The accessible velocity region is divided into two half planes by  $\mathbf{v}_A(t)$ . Two vectors, the closest vectors from  $\mathbf{v}_{des}(t)$  outside of the velocity obstacle region, are selected in the left half plane and the right half plane. The two vectors are represented as  $\mathbf{v}_{LHP}$  and  $\mathbf{v}_{RHP}$  respectively. The suitability of the two selected vectors is evaluated through following proposed evaluation functions.

$$C_L = \left[ 1 + \left\{ \sum_{i=1}^n \left( \frac{1}{\bar{R}_{iA}^{rot}} \right) - \frac{1}{R_{goalA}^{rot}} \right\} \right] \cdot \left\| \begin{aligned} &\alpha \cdot (\mathbf{v}_{LHP}(t) - \mathbf{v}_{des}(t)) \\ &+ (\mathbf{v}_{LHP}(t) - \mathbf{v}_A(t)) \\ &+ (\mathbf{v}_{LHP}(t) - \mathbf{v}_{max} \cdot e^{i^0}) \end{aligned} \right\| \quad (4.15)$$

$$C_R = \left[ 1 - \left\{ \sum_{i=1}^n \left( \frac{1}{\bar{R}_{iA}^{rot}} \right) - \frac{1}{R_{goalA}^{rot}} \right\} \right] \cdot \left\| \begin{aligned} &\alpha \cdot (\mathbf{v}_{RHP}(t) - \mathbf{v}_{des}(t)) \\ &+ (\mathbf{v}_{RHP}(t) - \mathbf{v}_A(t)) \\ &+ (\mathbf{v}_{RHP}(t) - \mathbf{v}_{max} \cdot e^{i^0}) \end{aligned} \right\| \quad (4.16)$$

In the above equations,  $\bar{R}_{iA}^{rot}$  is the average of the time-varying radii of the rotation for the obstacle  $i$ , calculated by equation (4.5).  $\bar{R}_{iA}^{rot}$  represents the configuration of the velocity obstacle region in robot-centered coordinates. The sign of  $\bar{R}_{iA}^{rot}$  signifies the left and right side of the robot, and the magnitude of  $\bar{R}_{iA}^{rot}$  is associated with the relation between the robot and the obstacle. When the magnitude of  $\bar{R}_{iA}^{rot}$  is small, it means that the obstacle is very close to the left or right side of the robot. In this case, the robot should choose the opposite side as its new direction to avoid a collision even if it is little bit farther. The first term of the evaluation function is designed to lead the robot away from the obstacle which has small radius of rotation.

However, the robot should approach to the goal when the goal point is near the robot, although several obstacles also exist near the goal point. To attract the robot to the destination,  $R_{goalA}^{rot}(t)$  is subtracted from the summation term. The braces of the evaluation function are limited to between -1 and 1. The limitation balances the effect of the first and second term of the evaluation function. When the obstacle is placed in the forward direction of the robot, the magnitude of  $\bar{R}_{iA}^{rot}$  is large and the first terms of both evaluation functions become almost the same. In this case, the robot has a choice in avoidance direction. The second term of the evaluation function, the absolute value, helps to choose new direction of the robot.

The first element of the absolute value term leads the robot toward the goal. Because the velocity obstacle regions are bunched near the horizontal axis of robot-centered polar coordinates, the proposed algorithm occasionally guides the robot along the farther boundary of the velocity obstacle regions. In this case, the robot may wander around the goal point. To prevent wandering motion, a constant parameter  $\alpha$  is multiplied to the first element of the absolute value term. The parameter  $\alpha$  is designed to lead the robot to the destination more strongly. Large value of  $\alpha$  drives the robot directly to the destination.

The middle part of the absolute value term prevents sudden changes of direction of the robot, and the last part is designed to reduce zigzag motion and traveling time of the robot. Angular velocity which has the opposite sign

to the current angular velocity causes zigzag motion of the robot. Therefore,  $\mathbf{v}_{LHP}$  and  $\mathbf{v}_{RHP}$  can be evaluated by a vector which exists on the horizontal axis of robot-centered polar coordinates. If the desired velocity  $\mathbf{v}_{des}(t)$  overlaps with the velocity obstacle region, a vector which is in opposite direction can be the nearest one. When the robot selects this vector without any further consideration, it may repeat changing its direction forever. The last element of the evaluation function prevents this situation. To drive the robot to the goal point as fast as possible, the magnitude of the last component is determined to  $v_{max}$ .

The robot evaluates two alternative velocities and calculates the final evaluation value as equation (4.17). Then, the robot chooses new velocity for next time step as presented in equation (4.18).

$$C = \text{round}\left(\frac{C_R}{C_R + C_L}\right) \quad (4.17)$$

$$\mathbf{v}_A(t + \Delta t) = C \cdot \mathbf{v}_{LHP}(t) + (1 - C) \cdot \mathbf{v}_{RHP}(t) \quad (4.18)$$

## 4.5 Kinematic Constraints and Execution

In this dissertation, the proposed algorithm is applied to a mobile robot with differential drive constraints. As represented in [29], a simple kinematic model of the differential drive robot is constructed as follows:

$$\dot{x} = \frac{v_l + v_r}{2} \cos \theta \quad (4.19)$$

$$\dot{y} = \frac{v_l + v_r}{2} \sin \theta \quad (4.20)$$

$$\dot{\theta} = \frac{v_r - v_l}{L}. \quad (4.21)$$

The speeds of the right and left wheel are  $v_r$  and  $v_l$  respectively, and the distance between two wheels is  $L$ . The above equations can be rewritten in the form of polar coordinates, and then the speeds which are transmitted to each wheel are rewritten as below:

$$v_r = \frac{L\dot{\theta}}{2} + \dot{r} \quad (4.22)$$

$$v_l = -\frac{L\dot{\theta}}{2} + \dot{r}. \quad (4.23)$$

If the magnitude of  $v_r$  or  $v_l$  is larger than  $v_{max}$ , both  $v_r$  and  $v_l$  are adjusted less than  $v_{max}$  in the same proportion. The adjusted  $v_r$  and  $v_l$  satisfy the kinematic constraints. Then,  $\mathbf{v}_A(t + \Delta t)$  which generates  $v_r$  and  $v_l$  changes the robot's position as equation (4.24). At next time step  $t + \Delta t$ , the robot repeats the same process after resetting the origin point and the orientation angle as the robot's new position. The whole procedure in the environments with multiple obstacles is presented in Table 4.1.

$$\begin{aligned} \mathbf{p}_A^{polar}(t + \Delta t) &= \mathbf{p}_A^{polar}(t) + \int_t^{t+\Delta t} \mathbf{v}_A(\tau) d\tau \\ &= \mathbf{p}_A^{polar}(t) + \int_t^{t+\Delta t} \dot{r}_A(\tau) \cdot e^{i\dot{\theta}_A(\tau)} d\tau \end{aligned} \quad (4.24)$$



TABLE 4.1  
ALGORITHM OF THE VELOCITY OBSTACLES IN POLAR COORDINATES FOR  
MULTIPLE OBSTACLES

<b>Input:</b>	List of relative positions of obstacles $\mathcal{O}$
<b>Output:</b>	Velocity of robot at next time step $\mathbf{v}_A(t + \Delta t)$
1:	Initially given $\mathbf{v}_A(0)$ and $\mathbf{p}_{goal}$
2:	$t=0$
3:	<b>while</b> ( $\ \mathbf{p}_{goalA}^{polar,rel}(t + \Delta t)\  > \epsilon$ ) <b>do</b>
4:	$t=t+\Delta t$
5:	<b>for all</b> ( $\mathcal{O}_j \in \mathcal{O}$ ) <b>do</b>
6:	Sense obstacle position $\mathbf{p}_{O_jA}^{polar,rel}(t)$ by updating input
7:	Calculate $\mathbf{v}_{O_j}(t)$
8:	Generate $\mathbf{v}_{des}(t)$ using (4.2) and (4.3)
9:	Generate $VOP_{O_jA}^K$ using (4.6), (4.8), and (4.9)
10:	<b>end for</b>
11:	<b>if</b> $\mathbf{v}_{des}(t)$ intersects with $\bigcup_{O_j \in \mathcal{O}} (VOP_{O_jA}^K)$ <b>then</b>
12:	Calculate $\mathbf{v}_{LHP}$ and $\mathbf{v}_{RHP}$
13:	Calculate $C_L$ and $C_R$ , evaluations of $\mathbf{v}_{LHP}$ and $\mathbf{v}_{RHP}$
14:	Calculate decision value $C$ by (4.13)
15:	Determine $\mathbf{v}_A(t + \Delta t)$ by (4.14)
16:	<b>else</b>
17:	$\mathbf{v}_A(t + \Delta t) \leftarrow \mathbf{v}_{des}(t)$
18:	<b>end if</b>
19:	<b>return</b> $\mathbf{v}_A(t + \Delta t)$
20:	<b>end while</b>

## Chapter 5

# Analysis of Velocity Obstacles in Polar Coordinates

### 5.1 Collision-free Navigation

In this section, it is mathematically proved that the velocity vector outside the velocity obstacle region of the proposed algorithm guarantees collision-free with respect to the obstacle. To prove collision-free, robot velocity which leads collisions with the obstacle is analyzed with formulaic representation in robot-centered polar coordinates.

Since the robot perceives the current position and velocity of the obstacle, it can estimate the position of the obstacle at time  $\kappa$  as described in Figure 4.3. The robot has piecewise constant linear and angular velocities, therefore, the robot moves with uniform circular motion by maintaining velocity (equation

(4.6) and (4.8)) on condition that external factors do not affect. If the robot is heading for the center position of the obstacle at time  $\kappa$ , then the robot follows its trajectory as described in Figure 4.3. If the robot moves toward the outer boundary of the obstacle, the robot has its trajectory in accordance with the crossing point of the robot trajectory and the obstacle boundary, and the robot trajectories are represented in Figure 5.1. Because the initial orientation angle of the robot is 0radian, the robot trajectories passing the obstacle boundary are parts of circles which have the center positions on the vertical axis of the polar coordinates. Among the circular trajectories, the circle which is circumscribed on the obstacle boundary becomes the robot trajectory on the left end, and the circle which is inscribed on the obstacle boundary becomes the right end robot trajectory as shown in Figure 5.2.

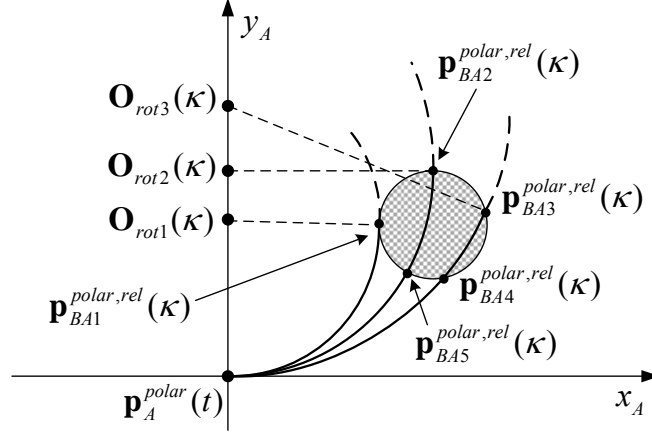


Figure 5.1 The crossing points of the robot trajectory and the obstacle boundary at time  $\kappa$ .

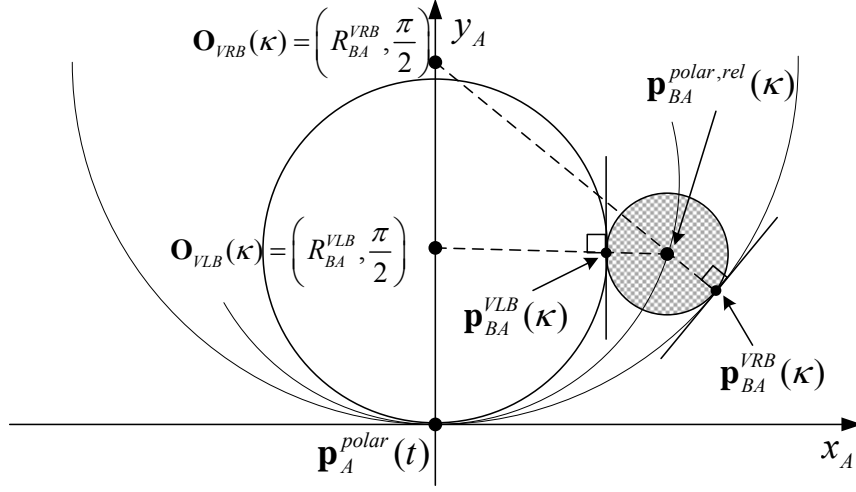


Figure 5.2 The robot trajectories passing left and right end of the boundary of the obstacle. The two robot trajectories share common tangent lines with the obstacle boundary.

The radius of rotation of the robot, enforcing the robot toward the contact point of the left end trajectory  $\mathbf{p}_{BA}^{VLB}(\kappa)$ , is calculated as equation (5.1) by using the characteristics of two circumscribed circles and the law of cosines to  $\Delta \mathbf{p}_A^{polar}(t) \mathbf{p}_{BA}^{polar,rel}(\kappa) \mathbf{O}_{VLB}(\kappa)$ .

$$R_{BA}^{VLB}(\kappa) = \frac{\rho_{BA}^{rel}(\kappa)^2 - R_{BA}^2}{2(\rho_{BA}^{rel}(\kappa) \cdot \sin(\phi_{BA}^{rel}(\kappa)) + R_{BA})} \quad (5.1)$$

When the law of cosines is applied to  $\Delta \mathbf{p}_A^{polar}(t) \mathbf{p}_{BA}^{VLB}(\kappa) \mathbf{O}_{VLB}(\kappa)$  with respect to  $\angle \mathbf{p}_A^{polar}(t) \mathbf{O}_{VLB}(\kappa) \mathbf{p}_{BA}^{VLB}(\kappa)$ , the orientation angle of  $\mathbf{p}_{BA}^{VLB}(\kappa)$  is

calculated as below:

$$\phi_{BA}^{VLB}(\kappa) = \frac{1}{2} \cos^{-1} \left\{ \frac{\left( R_{BA}^{VLB}(\kappa) + R_{BA} \right)^2 + R_{BA}^{VLB}(\kappa)^2 - \rho_{BA}^{rel}(\kappa)^2}{2 \cdot R_{BA}^{VLB}(\kappa) \cdot \left( R_{BA}^{VLB}(\kappa) + R_{BA} \right)} \right\}. \quad (5.2)$$

To arrive  $\mathbf{p}_{BA}^{VLB}(\kappa)$  at time  $\kappa$  with the constant radius of rotation  $R_{BA}^{VLB}(\kappa)$ , the robot has constant angular velocity  $\dot{\theta}_{BA}^{VLB}(\kappa)$  which changes the orientation of the robot  $2\phi_{BA}^{VLB}(\kappa)$  during  $\kappa$ . Linear velocity of the robot is determined upon the angular velocity value, since the radius of rotation is fixed. Therefore, linear and angular velocities of the robot generating the left end trajectory at time  $\kappa$  are represented as equation (5.3).

$$\begin{aligned} \mathbf{v}_{BA}^{VLB}(\kappa) &= \left( \dot{r}_{BA}^{VLB}(\kappa), \dot{\theta}_{BA}^{VLB}(\kappa) \right) \\ &= \left( 2 \cdot R_{BA}^{VLB}(\kappa) \cdot \sin\left(\frac{\phi_{BA}^{VLB}(\kappa)}{\kappa}\right), \frac{2 \cdot \phi_{BA}^{VLB}(\kappa)}{\kappa} \right) \end{aligned} \quad (5.3)$$

Robot velocity of the right end trajectory at time  $\kappa$ ,  $\mathbf{v}_{BA}^{VRB}(\kappa)$ , is calculated in a similar way that is mentioned above. For the right end boundary velocity, the characteristics of two inscribed circles and the law of cosines are used, and the related equations are represented as equation (5.4) through (5.6).

$$R_{BA}^{VRB}(\kappa) = \frac{\rho_{BA}^{rel}(\kappa)^2 - R_{BA}^2}{2 \left( \rho_{BA}^{rel}(\kappa) \cdot \sin(\phi_{BA}^{rel}(\kappa)) - R_{BA} \right)} \quad (5.4)$$

$$\phi_{BA}^{VRB}(\kappa) = \frac{1}{2} \cos^{-1} \left\{ \frac{\left( R_{BA}^{VRB}(\kappa) - R_{BA} \right)^2 + R_{BA}^{VRB}(\kappa)^2 - \rho_{BA}^{rel}(\kappa)^2}{2 \cdot R_{BA}^{VRB}(\kappa) \cdot \left( R_{BA}^{VRB}(\kappa) - R_{BA} \right)} \right\} \quad (5.5)$$

$$\begin{aligned}
\mathbf{v}_{BA}^{VRB}(\kappa) &= \left( \dot{r}_{BA}^{VRB}(\kappa), \dot{\theta}_{BA}^{VRB}(\kappa) \right) \\
&= \left( 2 \cdot R_{BA}^{VRB}(\kappa) \cdot \sin\left(\frac{\phi_{BA}^{VRB}(\kappa)}{\kappa}\right), \frac{2 \cdot \phi_{BA}^{VRB}(\kappa)}{\kappa} \right)
\end{aligned} \quad (5.6)$$

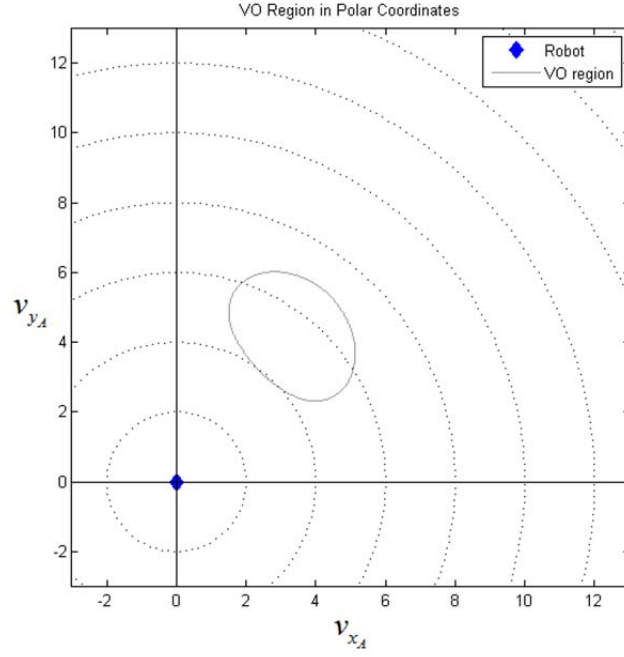


Figure 5.3 The velocity obstacle region at time  $\kappa$   $\{\mathbf{VOP}_{AB}^\kappa\}$  (black closed curve) on polar coordinates of robot velocity.

Since piecewise linearity is established for linear and angular velocities of the robot, one to one correspondence is valid between robot velocities and uniform circular trajectories. Therefore, the robot cannot have trajectories out of the region between the left and right end robot trajectories to reach a

particular point on the obstacle boundary (e.g.  $\mathbf{p}_i(\kappa)$ ,  $i=1$  to 5, of Figure 5.1). Robot velocities toward the every point of the obstacle boundary are calculated in a similar way to  $\mathbf{v}_{BA}^{VLB}(\kappa)$  and  $\mathbf{v}_{BA}^{VRB}(\kappa)$ , and the result velocities form a velocity obstacle region at time  $\kappa$   $\{VOP_{AB}^\kappa\}$ .  $\{VOP_{AB}^\kappa\}$  is represented as a black closed curve in Figure 5.3 in robot-centered polar coordinates of robot velocity. If the robot has velocity inside of the closed contour, then the robot A collides with the obstacle B after  $\kappa$  time.

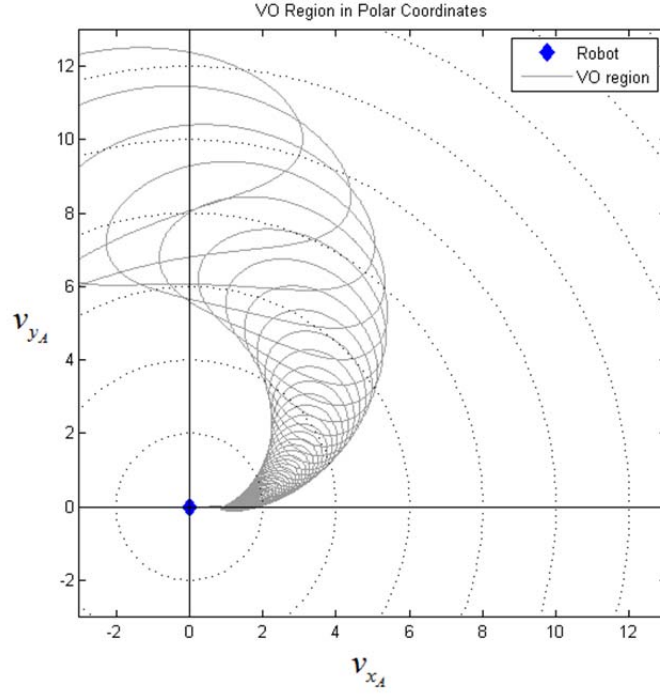


Figure 5.4 The velocity obstacle region for the obstacle B  $\{VOP_{AB}\}$  on polar coordinates of robot velocity.  $\{VOP_{AB}\}$  is a union of  $\{VOP_{AB}^\kappa\}$ .

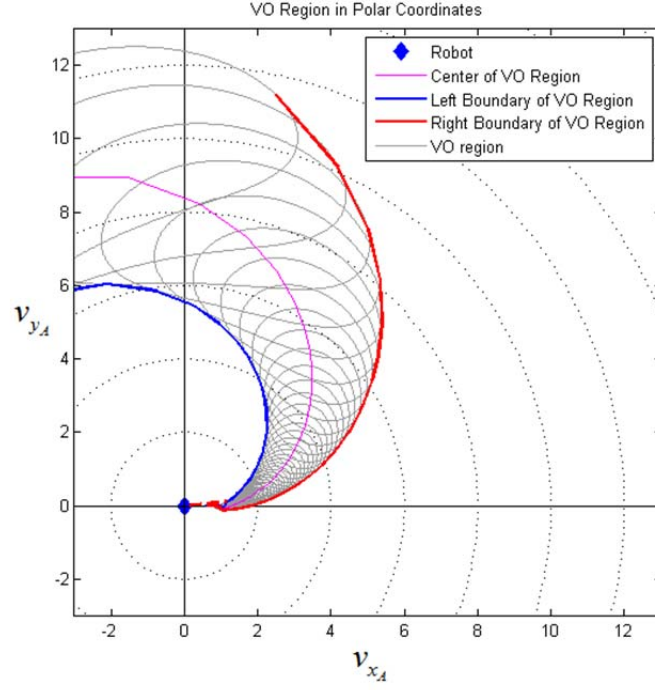


Figure 5.5 The velocity obstacle region  $\{\mathbf{VOP}_{AB}\}$  and the left and right end boundaries of the velocity obstacle region. The left end boundary of the  $\{\mathbf{VOP}_{AB}\}$  is a blue curve, and the right end boundary is a red curve. Robot velocities which lead the robot to the center of the obstacle are represented as a purple curve.

By calculating time-varying  $\{\mathbf{VOP}_{AB}^\kappa\}$ , the total velocity obstacle region  $\{\mathbf{VOP}_{AB}\}$  can be obtained as a union of  $\{\mathbf{VOP}_{AB}^\kappa\}$  as shown in Figure 5.4. The outer boundary of  $\{\mathbf{VOP}_{AB}\}$  is generated by accumulating  $\mathbf{v}_{BA}^{VLB}(\kappa)$  and  $\mathbf{v}_{BA}^{VRB}(\kappa)$  which are two end velocities of  $\{\mathbf{VOP}_{AB}^\kappa\}$ , and it consists of two simple curves. The left and right end boundaries of robot velocity are



represented as blue and red curves of Figure 5.5. The two velocities of the left and right boundaries, expressed as equation (5.3) and (5.6), consist of terms related to velocity of the obstacle. Therefore, the two end boundary velocities can be defined as follows:

$$\mathbf{v}_{BA}^{VLB}(\kappa) = f(\mathbf{v}_B(\tau)) \quad (5.7)$$

$$\mathbf{v}_{BA}^{VRB}(\kappa) = g(\mathbf{v}_B(\tau)) . \quad (5.8)$$

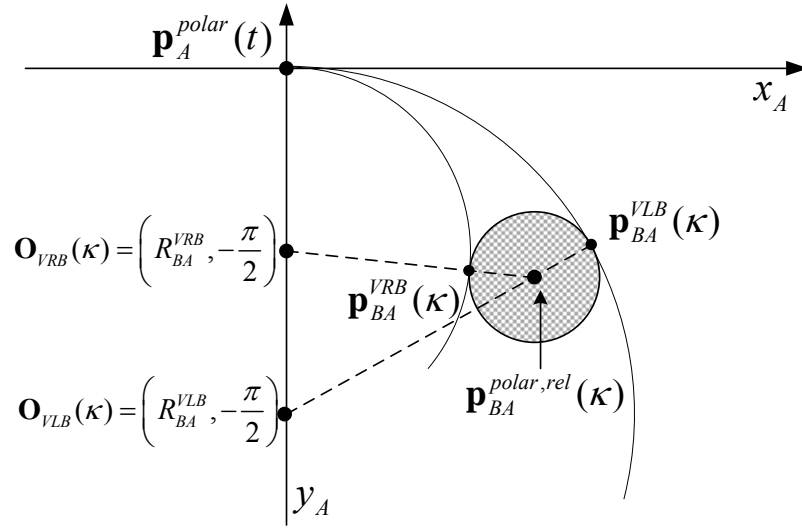


Figure 5.6 The robot trajectories passing left and right end of the boundary of the obstacle on the right side of the robot. The circumscribed circle is the right end robot trajectory, and the inscribed circle is the left end robot trajectory in this case.

At the new velocity decision process, the robot chooses its new velocity outside of the velocity obstacle region. Consequently, it is guaranteed that the

robot navigates with collision-free path using the proposed algorithm of this dissertation. Furthermore, the velocity obstacle region generation process of this dissertation automatically reflects velocity of the obstacle in contrast with the conventional velocity obstacle approaches.

When the obstacle is placed in the right side of the robot as shown in Figure 5.6, the circumscribed circle becomes the left end trajectory and the inscribed circle becomes the right end trajectory of the robot.

## 5.2 Smoothness of the Robot Trajectory

In mathematical analysis, a function is called smooth, which has derivatives of all orders. Especially for two-dimensional segments (e.g. curves), the function is smooth when the second derivatives are continuous in the entire segments. Geometric continuity [30, 31] is one of the concepts to represent smoothness of a curve of surface. The basic idea of geometric continuity was primarily proposed to interpret continuity between the various sections of the conic. Geometric continuity is extended for an intrinsic measure of continuity appropriate for spline development. Geometric continuity is a relaxed form of parametric continuity independent of the parameterizations of the curve segments.

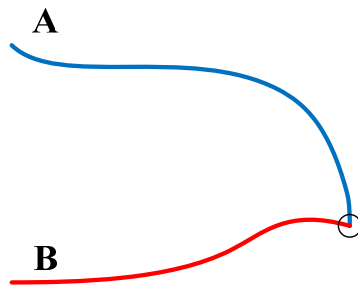
Geometric continuity is based on arc-length parameterizations, and it can be shown that two parameterizations meet with  $G^n$  continuity if and only if the corresponding arc-length parameterizations meet with  $C^n$  parametric

continuity [31]. A re-parameterization of the curve is geometrically identical to the original. Then, a curve or surface can be described as having  $G^n$  continuity, and high order  $n$  means smoothness increase.  $G^n$  continuity can be stated formally as a below theorem. Figure 5.7 represents examples of  $G^n$  continuity.

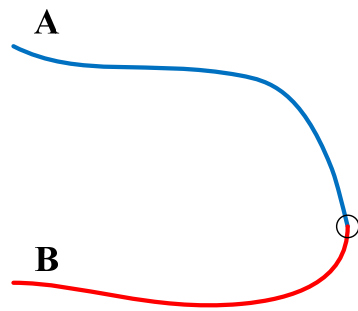
- $G^0$  continuity: The segments of the curve touch at the join point.
- $G^1$  continuity: The segments also have a common unit tangent vector at the join point, and satisfy  $G^0$  continuity.
- $G^2$  continuity: The segments also have a common center of curvature at the join point, and satisfy  $G^1$  continuity.

As shown in Figure 5.7, typically, a curve which has  $G^1$  continuity is enough to appear smooth. However, a trajectory of the robot is affected by changes of robot acceleration. Therefore, robot acceleration should be continuous in every segment of the robot trajectory. In this reason,  $G^2$  continuity of the curve is important for the robot trajectory.

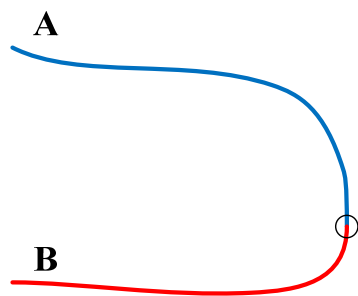
In this dissertation, the proposed algorithm generates a uniform circular trajectory of the robot as stated in Section 4.2. If any obstacle does not exist near the robot, the proposed algorithm calculates a radius of rotation of the robot to the destination by equation (4.2), and then calculates linear and angular robot velocities to maintain the radius of rotation by equation (4.3). The radius of rotation determines curvature value of the robot trajectory.



(a)  $G^0$  continuity



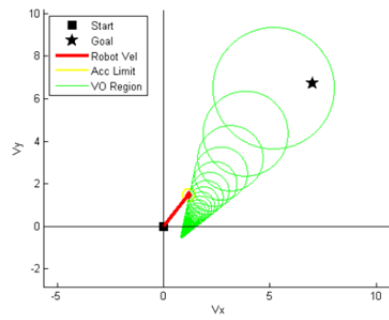
(b)  $G^1$  continuity



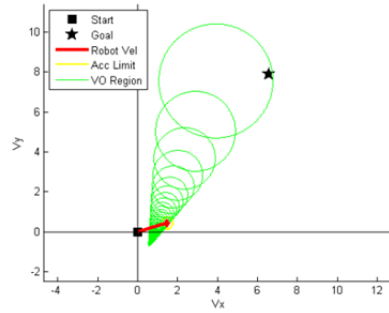
(c)  $G^2$  continuity

Figure 5.7 Examples of geometric continuity of a two-dimensional curve. The two segments of the curve are  $G^n$  continuous: (a) the two segments with  $G^0$  continuity, (b) the two segments with  $G^1$  continuity, (c) the two segments with  $G^2$  continuity.

Uniform circular motion is established by maintaining curvature value, and curvature of the robot trajectory is constant when robot acceleration is constant. In other words, the robot using the proposed algorithm inherently has a constant acceleration in the non-obstacle condition, and follows the trajectory with  $G^2$  continuity.



(a) the case of choosing the left side of the velocity obstacle region



(b) the case of choosing the right side of the velocity obstacle region

Figure 5.8 The velocity selection results of the conventional velocity obstacle algorithm causing oscillations. Figures are captured from MATLAB simulations: (a) the selected velocity on the left side of the velocity obstacle region, (b) the selected velocity on the right side of the velocity obstacle region at the next time step.

When the robot meets obstacles, the robot cannot retain the previous uniform circular motion. In this case, most of the conventional collision avoidance algorithms calculate new velocity vector with the minimum Euclidean distance from current velocity. The conventional methods bring minimum acceleration changes, so it looks good for instant reaction. However, the performance evaluation indices of the robot trajectory are very easy to get lower. Because the conventional methods do not consider any further information about the future situation, oscillation of the trajectory is generated occasionally as shown in Figure 5.8.

As described in Section 4.4, the proposed algorithm of this dissertation is designed to reflect estimations for future situations to the velocity selection process. The new algorithm selects two candidate velocities, and then evaluates using the evaluation function. The evaluation function considers four evaluation factors: keeping away from the obstacle, approaching to the destination, reducing acceleration changes, and decreasing rotational motion. Because of the first evaluation factor of the proposed evaluation function, the proposed algorithm can reduce oscillated robot motion. Naturally, geometric continuity can be well-established in entire time of the robot navigation.

### **5.3 Local Minima Avoidance**

Researches on autonomous robot navigation in unknown environments

have been subject to local minima problems. In mathematics, the local minimum is the smallest value that a function takes at a point within a given neighborhood: local extremum of the function. In the reactive navigation field, the local minimum means the situation that the robot cannot travel to appropriate direction because of its local sensory information. Since the robot only uses local information, the obstacle avoidance algorithm may calculate wrong direction which is from local extrema of the heuristic path planning function.

There are methods in literature that tackle the local minima problem such as the Bug algorithms [32, 33, 34, 35], potential field [36], and their recent improvements [37, 38, 39, 40, 41]. Furthermore, significant efforts have been dedicated to overcome this problem, often by using approaches from other disciplines of study. [42] used harmonics functions from fluid dynamics and [43] used Maxwell's equations. Other examples have been studied for obstacle avoidance navigation including approaches such as discrete grid based [44] and central path computation [45]. Unfortunately, most of the randomized or optimization driven path planning algorithms is expensive in particular environments, and may even fail to reach the destination. Many navigation algorithms which have been studied to solve the local minima problem work in the particular conditions or with the heuristic functions.

The local minima problem is inevitable for reactive navigation in unknown dynamic environments, since the robot cannot predict local minima

before it detects obstacles causing local extremum of the avoidance algorithm. As mentioned in [14], the local minima problem is also inevitable for the velocity obstacle algorithm. Because of the hereditary characteristics, the local minima problem cannot be totally solved. However, in this dissertation, factors which affect the local minima situation are analytically studied to reduce the local minima occurrences.

The local minima problems can be occurred when the situation is ambiguous to decide proper direction to avoid collisions, velocity of the obstacle is unpredictable, or new obstacle is detected. The oscillation situation in Figure 5.8 is happened because there are no additional criteria for velocity selection when several available velocities are in similar conditions. If the robot selects the local minimum repeatedly, the robot cannot conduct obstacle avoidance properly just before conflicts as shown in Figure 5.9. Most of the conventional velocity obstacle algorithms consider linear velocity only, so obstacle motion with angular velocity is not reflected in the velocity decision process. Therefore, the accessible velocity space of the robot cannot exist by surrounding velocity obstacle regions of several obstacles. The situation is called the stuck situation, and it is described in Figure 5.10. When new obstacle is detected within the sensing range, the stuck situation can be occurred. However, this case is unavoidable without communication with other robot or systems, or sensor range extension.



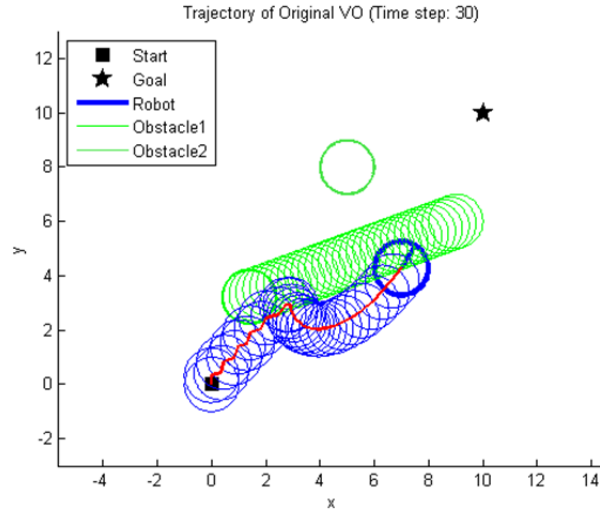
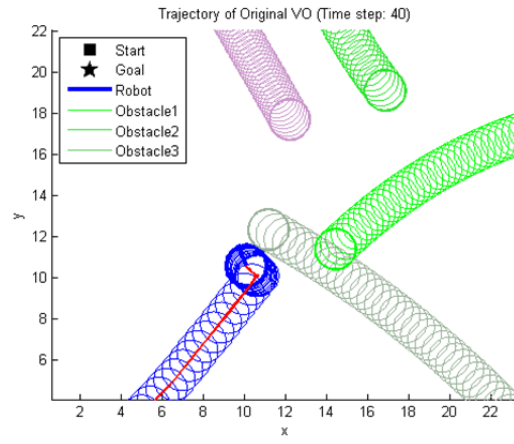
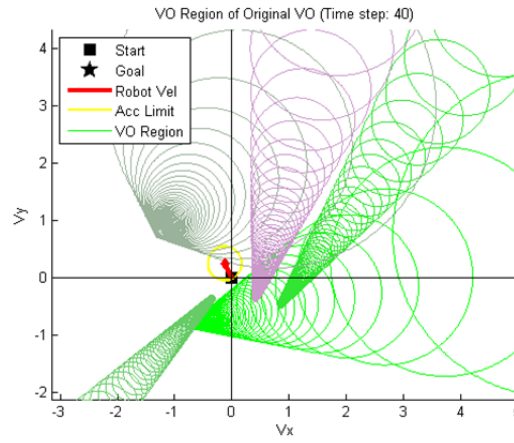


Figure 5.9 The robot trajectory with oscillation and the local minima problem. After iterating several oscillated motion, the robot faces the local minima situation. The robot barely avoids collisions with the obstacle.

The proposed algorithm of this dissertation reduces the local minima problem by parameterizing information of surrounding environments of the robot. As confirmed in the above section, the proposed algorithm evaluates alternative velocities from both sides of the accessible velocity region by the evaluation function. Figure 5.11 shows the correspondences between four evaluation factors and each term of the evaluation function. Some of the evaluation factors are containing information related with the local minima problem. One of the main reasons of local minima is oscillated motion, so oscillation reduction generated in the above section also affects the local minima avoidance.



(a) the robot and the obstacles trajectories in the stuck situation



(b) the original velocity obstacle regions in the stuck situation

Figure 5.10 The stuck situation due to the local minima problem: (a) the robot trajectory and obstacles trajectories with non-zero angular velocities, (b) the velocity obstacle regions of the original velocity obstacle algorithm in the same situation.

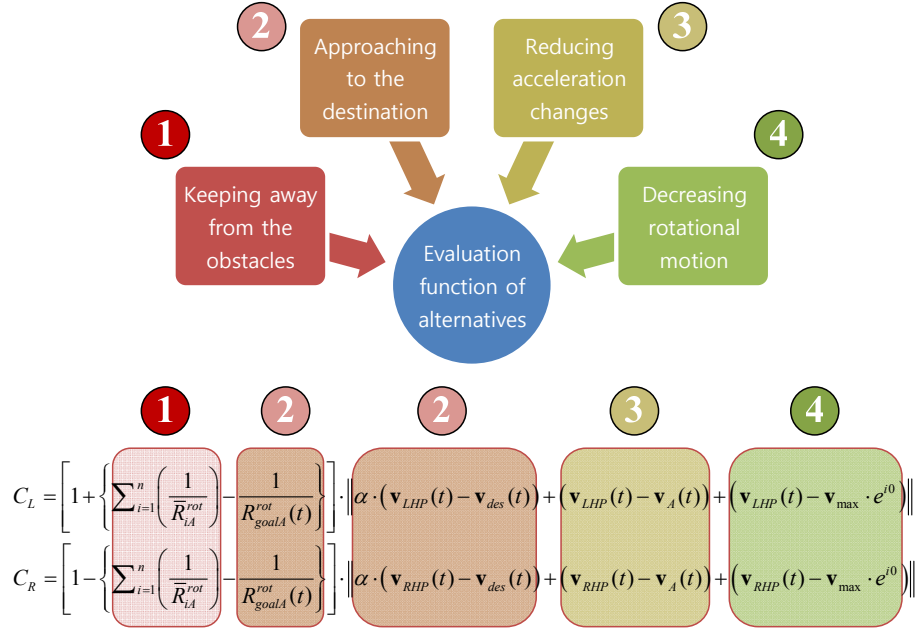


Figure 5.11 The four evaluation factors and their corresponding terms of the proposed evaluation functions. The evaluation function helps preventing oscillations and local minima of the robot motion.

The first term of the evaluation function affects the local minima prediction as well as the oscillation prevention. The obstacle motion with linear and angular velocities is reflected to the radius of rotation toward the center of the obstacle B,  $R_{BA}^{rot}(\tau)$ , at every time step. Accumulation of  $R_{BA}^{rot}(\tau)$  is the first term of the evaluation function. The sign of the first term signifies direction of the obstacle location. If the absolute magnitude of the first term of the evaluation function is large, then the corresponding velocity has low chances to be chosen. As the obstacle comes closer to the robot,

$R_{BA}^{rot}(\tau)$  is getting larger, so the first term is getting smaller. The obstacle which comes closer to the robot because of angular velocity results in increase in  $R_{BA}^{rot}(\tau)$ , as a result the first term of the evaluation function decreases. If angular velocity of the obstacle makes the obstacle move far away from the robot, the first term is large even though the obstacle has the same linear velocity with the former situation. In conclusion, the proposed algorithm predicts changes of obstacle motion more accurately than the conventional velocity obstacle algorithm; therefore the local minima problem can be reduced.

Other factors of the evaluation function help the robot reducing local minima in ambiguous situation to select its direction as in Figure 5.8. The conventional velocity obstacle algorithms consider only the second factor of the evaluation function, because they choose the velocity vector which has the smallest difference from the desired velocity vector. However, the robot can decide its direction consistently by considering the current robot motion and continuity of the motion.

Table 5.1 shows analysis of the proposed evaluation function. The components, reasons why the components are used, and the effects of each evaluation factor of the evaluation function are described. Every evaluation factor lowers the evaluation function value when the situation is pertinent to each factor. The components are used for the purposed of each evaluation factor.

TABLE 5.1

## ANALYSIS OF THE PROPOSED EVALUATION FUNCTION

Evaluation factors	Analysis section	Descriptions
Keeping away from the obstacles	Components	<ul style="list-style-type: none"> <li>- Average of the radius of rotation for obstacle <math>i</math> from time <math>t</math> to <math>\kappa</math>:  <math display="block">\bar{R}_{iA}^{rot} = \frac{1}{\kappa - t} \int_t^\kappa R_{iA}^{rot}(\tau) d\tau</math> </li> </ul>
	Reasons for use	<ul style="list-style-type: none"> <li>- The radius of rotation for an obstacle implies obstacle pose and movement.</li> </ul>
	Effects	<ul style="list-style-type: none"> <li>- High value is assigned to an alternative vector which exists or passes near the robot during <math>(\kappa - t)</math>.</li> </ul>
Approaching to the destination	Components	<ul style="list-style-type: none"> <li>- The radius of rotation which guides the robot to the goal: <math>R_{goalA}^{rot}(t)</math></li> <li>- Difference between <math>\mathbf{v}_{des}(t)</math> and alternatives</li> </ul>
	Reasons for use	<ul style="list-style-type: none"> <li>- When the goal position is near the robot, the radius of rotation for the goal is small.</li> <li>- <math>\mathbf{v}_{des}(t)</math> leads the robot to the goal as fast as possible.</li> </ul>
	Effects	<ul style="list-style-type: none"> <li>- The robot moves to the goal as much as it can.</li> <li>- Leverage of this term on the evaluation function varies by <math>\alpha</math>.</li> </ul>
Reducing acceleration changes	Components	<ul style="list-style-type: none"> <li>- Difference between current <math>\mathbf{v}_A(t)</math> and alternatives</li> </ul>
	Reasons for use	<ul style="list-style-type: none"> <li>- Substantial amount of velocity change reduces smoothness of the robot path.</li> </ul>
	Effects	<ul style="list-style-type: none"> <li>- Frequent velocity changes are lightened.</li> <li>- An alternative which maintains high geometric continuity will be more influential in selecting velocity.</li> </ul>
Decreasing rotational motion	Components	<ul style="list-style-type: none"> <li>- Difference between maximum linear velocity and alternatives: <math>\mathbf{v}_{max} e^{t_0}</math></li> </ul>
	Reasons for use	<ul style="list-style-type: none"> <li>- By increasing linear velocity, the robot moves long distance in a short time.</li> <li>- An alternative which has large angle from the robot's heading direction causes sudden turning motion.</li> </ul>
	Effects	<ul style="list-style-type: none"> <li>- This term suppresses rotational movement, so oscillation of the robot path decrease.</li> <li>- The robot moves as far as it can in limited time.</li> </ul>

The proposed navigation algorithm inherently has the velocity obstacle regions around the horizontal axis of robot-centered polar coordinates. After a considerable time, the robot does not need to have new velocity which has a big difference with the current velocity to meet the obstacle, so angular velocity of the robot  $\dot{\theta}_A$  becomes very small a lot of time later. Due to this inherent characteristic, the proposed navigation algorithm barely faces the stuck situation that the accessible velocity region is surrounded by the velocity obstacle region in the dynamic environments. However, the proposed algorithm is hard to find the shortest robot path between the crowded obstacles, so it tends to produce detour path of the robot, and we can find this tendency by simulations in the crowded environments. If period of time to calculate the velocity obstacle regions (e.g.  $\kappa$  of Section 4.3) is limited within particular time by considering velocities of the robot and the obstacle, the detour path of the robot might be shorten.

The proposed navigation algorithm of this dissertation has been tried to reduce the local minima problem by reflecting situation information to the velocity obstacle regions and the velocity decision process as stated above. However, the local minima problem cannot be totally solved in the inevitable extreme condition due to the inherent characteristic of the reactive navigation techniques. In this reason, the proposed algorithm is evaluated its performance of local minima avoidance compared to the conventional velocity obstacle algorithms by number of simulations.

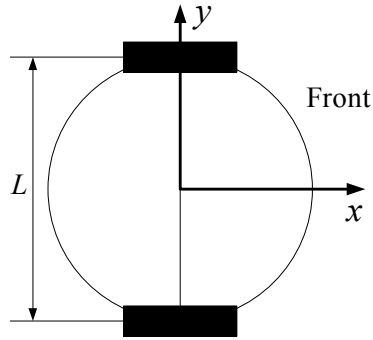
## Chapter 6

# Simulation Results

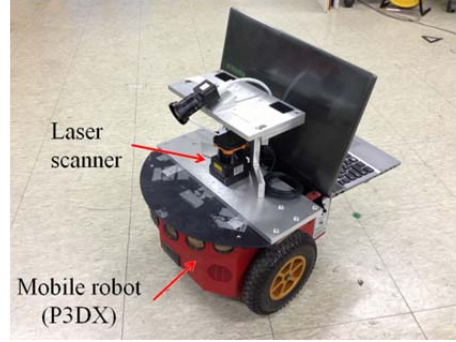
### 6.1 Implementation Setups

In this section, we describe the implementation of the velocity obstacle algorithm in robot-centered polar coordinates and compare results with other conventional algorithms. The robot is modeled as a two-wheeled differential drive robot such as Pioneer 3-DX in Figure 6.1. The distance  $L$  between two wheels of the robot is 40cm. The maximum speed of the robot  $v_{max}$  is 2.0m/s, and the maximum magnitude of the acceleration  $a_{max}$  is 1.0m/s<sup>2</sup>. The radii of the robot and obstacle are 1.0m. The maximum sensing range  $r_{range}$  is 10m. The parameter  $\alpha$  of the evaluation function is 1.0 in order to consider a balance between the every element of the evaluation function. All algorithms are performed every 0.3 seconds. The simulations are performed by MATLAB 2012 on a 3.4GHz Intel Core i7 CPU with 64bit Microsoft

Windows 7.



(a) Robot model



(b) Pioneer 3-DX with mounted sensors

Figure 6.1 The two-wheeled differential drive robot model for the simulations and the real robot model Pioneer 3-DX.

In order to fairly evaluate the performance of the obstacle avoidance algorithms, the same kinematic robot model is applied to all the algorithms. A method reflecting the kinematic constraints to the robot model from Section 4.5 is used for the simulations. When the new velocity of the robot by the conventional algorithm exceeds the limit of each wheel's velocity, the new velocity is adjusted in the same proportion.

The robot avoids multiple obstacles in different situations. The performances are evaluated by comparing the robot's total traveling time and length to the destination, and the computation time per iteration of each obstacle avoidance algorithm in the same space. Smoothness of the robot path is examined by measuring geometric continuity [31] of the robot trajectories,



since the curve, which is  $G^1$  and  $G^2$  continuous, does not have mechanical oscillations.

## 6.2 Simulations with a Single Dynamic Obstacle: Presence of Angular Velocity

As presented in Table 6.1, the initial position of the robot in global coordinates is (0,0) and the initial orientation angle is  $45^\circ$ . The robot has the initial velocity  $(\dot{r}_A(0), \dot{\theta}_A(0))$  as (1,0)(m/s,degree/s). The goal position in global coordinates is (10,10). A moving obstacle exists between the initial position of the robot and the goal. The performances of the collision avoidance algorithms depending on angular velocity are evaluated in two simple scenarios. The robot's trajectories are observed by increasing angular velocity of the obstacle.

TABLE 6.1  
INITIAL INFORMATION FOR SCENARIO 1 AND 2

Initial information (Common)	Robot position and orientation	$\mathbf{p}_A^{cart}(0) = (x_A(0), y_A(0), \theta_A(0)) = (0, 0, 45)$
	Robot velocity	$\mathbf{v}_A(0) = (\dot{r}_A(0), \dot{\theta}_A(0)) = (1, 0)$
	Goal position	$\mathbf{p}_{goal}^{cart}(0) = (x_{goal}, y_{goal}) = (10, 10)$
Scenario 1	Obstacle information	$\mathbf{p}_B^{cart}(0) = (4, 10, -40), \mathbf{v}_B(0) = (1, 0)$
Scenario 2	Obstacle information	$\mathbf{p}_B^{cart}(0) = (4, 10, -40), \mathbf{v}_B(0) = (1, -10)$

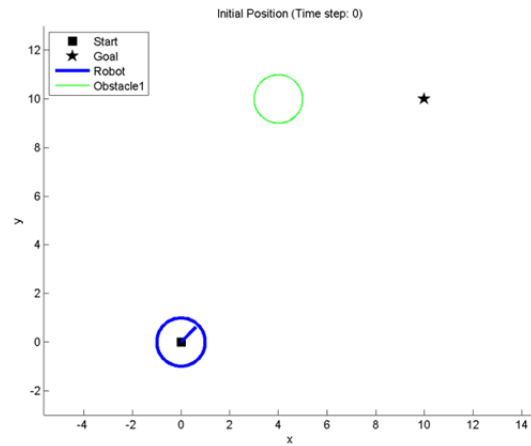
### 6.2.1 Scenario 1: An Obstacle with Zero Angular Velocity

Figure 6.2 (a) represents the initial situation of the first scenario, and Figure 6.2 (b) through (d) represent the final trajectories of the robot using the original velocity obstacles, the dynamic velocity space, and the proposed algorithm respectively. The situation is very simple. One obstacle is moving along a straight trajectory.

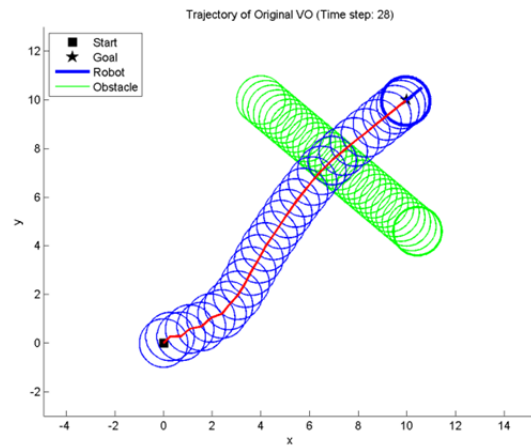
Overall, the shape of the robot trajectory of each algorithm looks similar, but the quality of smoothness is highly different. The robot using the original velocity obstacles has oscillations in the early part of its trajectory, and the robot using the dynamic velocity space has a drastic curve at the middle part of its travel as shown in Figure 6.2. The proposed navigation algorithm maintains robot velocity and acceleration as much as possible, so 86% of the robot trajectory satisfies  $G^1$  and  $G^2$  continuity as shown in Table 6.2. It is very high compared to other algorithms.

At the initial part, the velocity obstacle region of the original velocity obstacles is occupying a large area of the velocity space between the robot and the goal as in Figure 6.3 (a). Additionally,  $\mathbf{v}_{des}$  is around the middle of the velocity obstacle region, so the robot cannot decide which direction is proper to avoid collisions. As a result, the robot has a zigzag-shaped path, and runs appropriate action after oscillations. The dynamic velocity space algorithm selects the fastest velocity as the robot can achieve, so occasionally

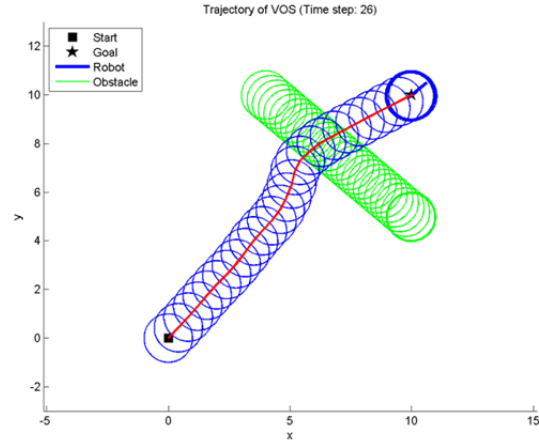
the robot moves too fast when the robot feels safe. In this reason, the drastic turning motion is occurred in the middle part of the robot trajectory. On the other hand, the proposed algorithm generates the velocity obstacle region bent to a particular direction, so the robot can choose the consistent direction.



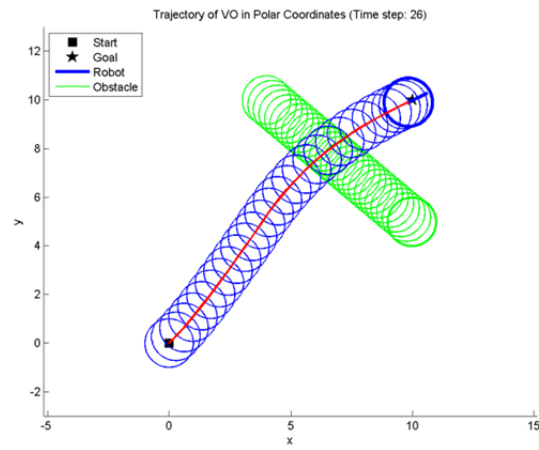
(a) Initial position



(b) Trajectory of VO

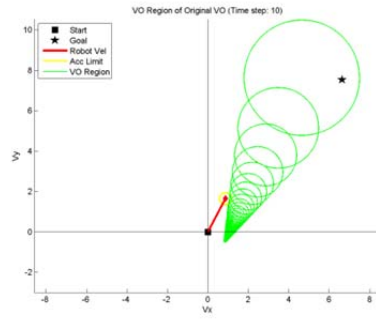


(c) Trajectory of DVS

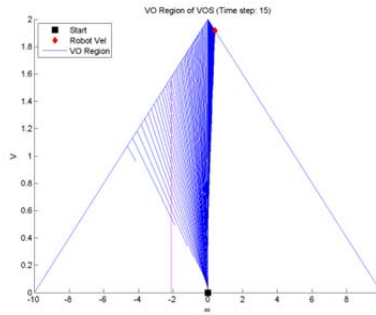


(d) Trajectory of VOP

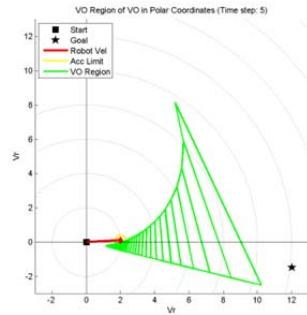
Figure 6.2 The global map and the robot trajectories of the first scenario with one moving obstacles with zero angular velocity. The mobile robot moves along the trajectories generated by the obstacle avoidance algorithms. The trajectory of the velocity obstacles shows zigzag motions and the trajectory of the dynamic velocity space has drastic turning motions. (VO: Velocity Obstacles, DVS: Dynamic Velocity Space, VOP: Velocity Obstacles in Polar coordinates)



(a) Velocity obstacle regions of VO



(b) Velocity obstacle regions of DVS



(c) Velocity obstacle regions of VOP

Figure 6.3 The velocity obstacle regions of three obstacle avoidance algorithms. The two conventional algorithms are drawn in Cartesian coordinates of robot velocity. The velocity obstacle regions of the proposed algorithm are drawn in robot-centered polar coordinates of velocity.

TABLE 6.2  
COMPARISON OF THE SIMULATION RESULTS OF SCENARIO 1

Algorithm	Total traveling time (s)	Total traveling distance (m)	Computation time per step (s)	Collisions (number)	Geometric continuity rate (%)
VO	8.1	14.49	0.0145	0	40.74
DVS	7.5	14.61	0.04	0	52.0
VOP	7.5	14.16	0.036	0	86.0

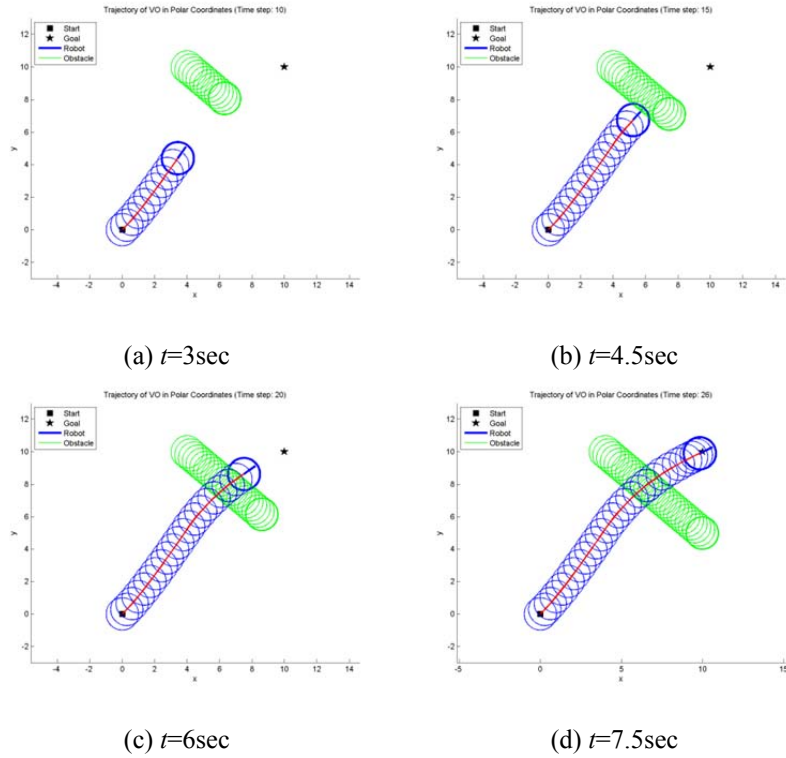
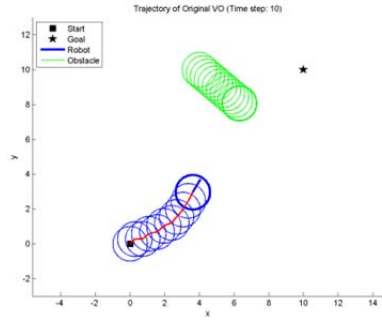
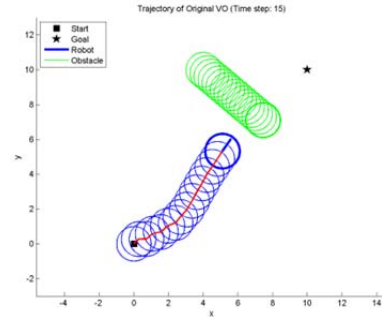


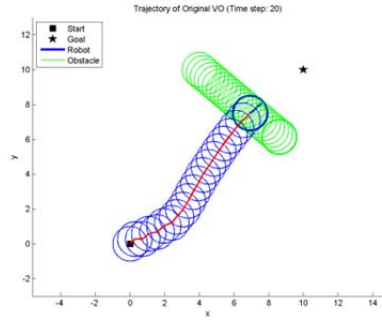
Figure 6.4 The trajectory of the robot using the proposed velocity obstacles in polar coordinates.



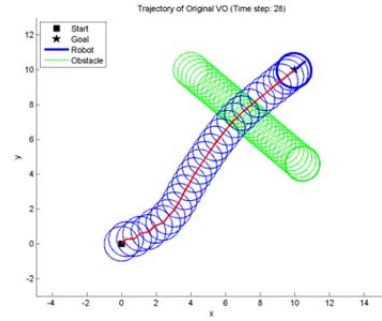
(a)  $t=3\text{sec}$



(b)  $t=4.5\text{sec}$



(c)  $t=6\text{sec}$



(d)  $t=8.1\text{sec}$

Figure 6.5 The trajectory of the robot using the original velocity obstacles over time. Oscillations are occurred until around  $t=3\text{sec}$ .

As presented in Table 6.2, the geometric continuity rate of the proposed algorithm is superior to other algorithms. The velocity obstacle region of the velocity obstacles in polar coordinates has a horn-shaped region as shown in Figure 6.3 (c), so the robot can have fast velocity which aims for the back of the obstacle. As a result, the robot using the proposed algorithm moves along

the shortest path in the quickest time as shown in Figure 6.4. The robot paths of other conventional algorithms are represented in Figure 6.5 and 6.6.

The computation time per iteration of the original velocity obstacles is shorter than the others. However, the computation time of the dynamic velocity space and the proposed algorithm is within the time interval, so the computation time of the both algorithms does not cause serious problems.

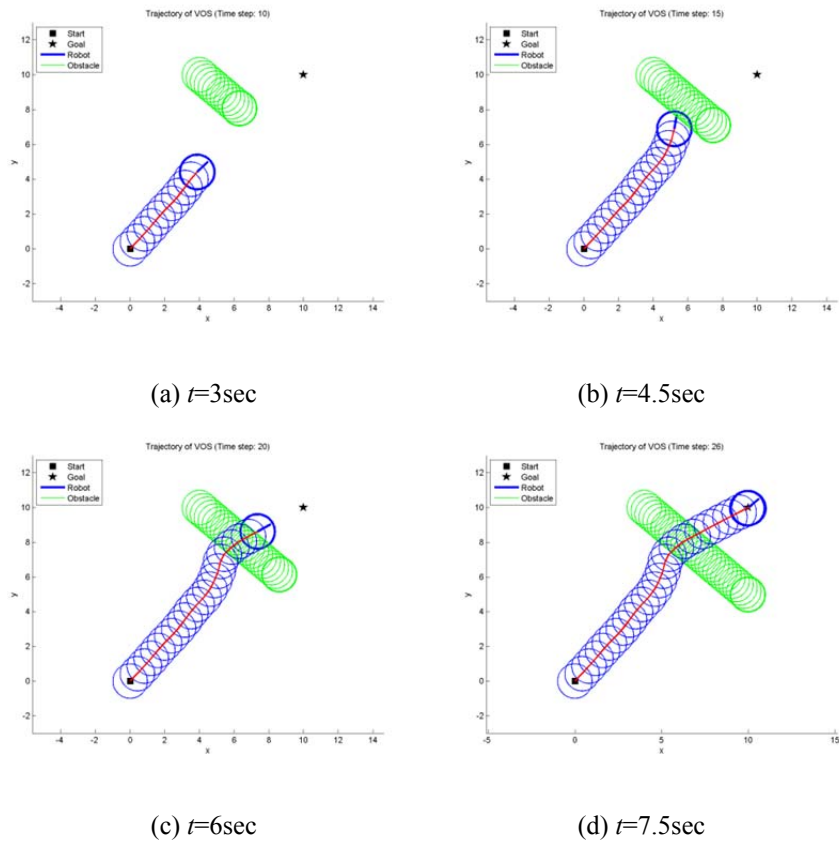


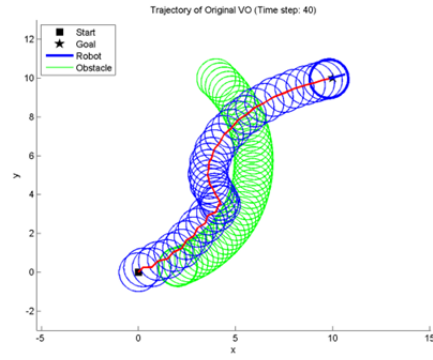
Figure 6.6 The trajectory of the robot using the dynamic velocity space over time. A drastic turning motion is occurred about  $t=5\text{sec}$ .



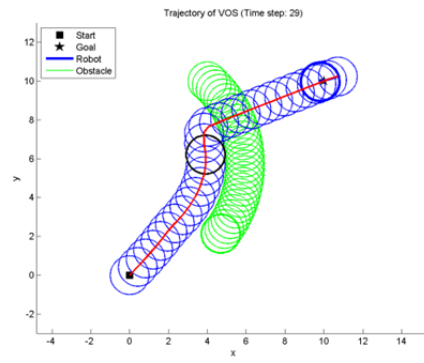
### **6.2.2 Scenario 2: An Obstacle with Non-Zero Angular Velocity**

The performance improvement is also noticeable in this scenario. The only difference from the above simulation is that the obstacle has non-zero angular velocity toward the robot. When the obstacle has angular velocity, the robot trajectory has more curved motion than the scenario 1 as in Figure 6.2. As presented in Figure 6.7, the shapes of the robot trajectories of the conventional velocity obstacle algorithms have more oscillations and drastic curves as the angular velocity of the obstacle becomes larger.

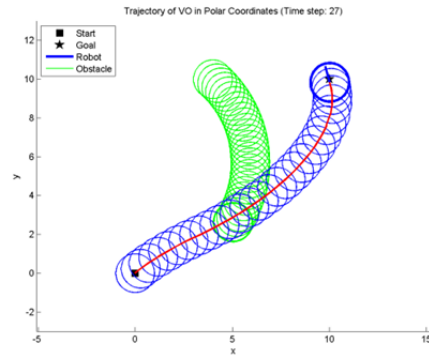
Since the conventional algorithms do not consider angular velocity of the obstacle, the future situation goes differently than the robot had expected. Therefore, the velocity obstacle regions do not guarantee collision avoidance, so oscillations and sharp curves are inevitable. However, the proposed algorithm reflects obstacle's linear and angular velocities at the velocity obstacle region generation process. Consequently, the robot maintains geometric continuity as much as possible. Geometric continuity of the dynamic velocity space increased as in Table 6.3, and it is because of increase in the straight forward motion of the robot. As a result, the robot should change direction suddenly about midway. In some unfortunate cases, the robot collides to the obstacle as in Figure 6.7 (b) (black circles).



(a) Trajectory of VO

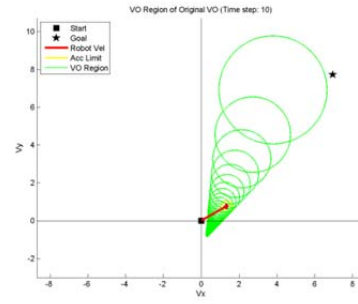
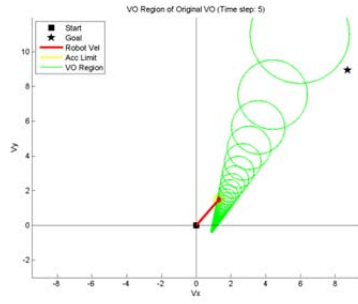


(b) Trajectory of DVS

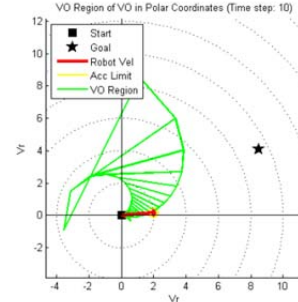
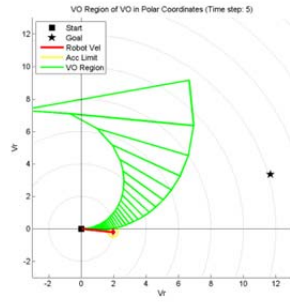


(c) Trajectory of VOP

Figure 6.7 The robot trajectories of the second scenario with one moving obstacles with non-zero angular velocity.



(a) Velocity obstacle region at  $t=1.5\text{sec}$  (b) Velocity obstacle region at  $t=3\text{sec}$



(c) Velocity obstacle region at  $t=1.5\text{sec}$  (d) Velocity obstacle region at  $t=3\text{sec}$

Figure 6.8 The velocity obstacle regions of the original velocity obstacles and the proposed algorithm at the initial part.

TABLE 6.3  
COMPARISON OF THE SIMULATION RESULTS OF SCENARIO 2

Algorithm	Total traveling time (s)	Total traveling distance (m)	Computation time per step (s)	Collisions (number)	Geometric continuity rate (%)
VO	11.7	16.16	0.0131	0	5.13
DVS	8.4	16.22	0.0386	2	64.29
VOP	7.8	14.95	0.0417	0	73.08

As shown in Figure 6.7 (c), the proposed navigation algorithm guides the robot to the right direction in contrast with other conventional algorithms. According to the shape of the velocity obstacle region and the evaluation function results, the robot prefers the alternative velocity on the right side of the velocity obstacle region as shown in Figure 6.8 (c). The velocity obstacle region contains information about rotation motion of the obstacle, and represents the information by flexing the contour of the velocity obstacle region. As a result, the robot chooses consistent direction as in Figure 6.8 (c) and (d). Consequently, the total traveling time and distance of the proposed algorithm are shorter than others, and the results are presented in Table 6.3. On the other hand, the original velocity obstacle algorithm cannot make the consistent decision about the heading direction at the initial part as represented in Figure 6.8 (a) and (b). Furthermore, the obstacle comes closer to the robot than expected, so the robot almost is caught in the stuck situation. The fastest computation time of the original velocity obstacles is not meaningful for this simulation.

### **6.3 Simulations with Multiple Dynamic Obstacles: Variation of Number of Obstacles**

In order to evaluate the performances of the collision avoidance algorithms in complex situation, they are evaluated by varying the number of the obstacles including the angular velocity in this section.

TABLE 6.4  
INITIAL INFORMATION FOR SCENARIO 3 AND 4

Initial information (Common)	Robot position and orientation	$\mathbf{p}_A^{cart}(0) = (x_A(0), y_A(0), \theta_A(0)) = (0,0,45)$
	Robot velocity	$\mathbf{v}_A(0) = (\dot{x}_A(0), \dot{\theta}_A(0)) = (1,0)$
Scenario 3	Goal position	$\mathbf{p}_{goal}^{cart}(0) = (x_{goal}, y_{goal}) = (10,10)$
	Obstacle information	$\mathbf{p}_B^{cart}(0) = (4,10,-40), \mathbf{v}_B(0) = (1,-2)$ $\mathbf{p}_C^{cart}(0) = (9,6,-160), \mathbf{v}_C(0) = (0.9,-1)$ $\mathbf{p}_D^{cart}(0) = (2,7,130), \mathbf{v}_D(0) = (1.1,0)$ $\mathbf{p}_E^{cart}(0) = (8,1,100), \mathbf{v}_E(0) = (0.4,2)$
Scenario 4	Goal position	$\mathbf{p}_{goal}^{cart}(0) = (x_{goal}, y_{goal}) = (30,30)$
	Obstacle information	$\mathbf{p}_B^{cart}(0) = (10,5,100), \mathbf{v}_B(0) = (0.5,0)$ $\mathbf{p}_C^{cart}(0) = (26,33,-80), \mathbf{v}_C(0) = (0.5,0)$ $\mathbf{p}_D^{cart}(0) = (25,13,130), \mathbf{v}_D(0) = (0.5,-0.6)$ $\mathbf{p}_E^{cart}(0) = (27,26,180), \mathbf{v}_E(0) = (0.5,0)$ $\mathbf{p}_F^{cart}(0) = (0,30,-30), \mathbf{v}_F(0) = (0.5,0)$ $\mathbf{p}_G^{cart}(0) = (20,25,-110), \mathbf{v}_G(0) = (0.5,0.6)$ $\mathbf{p}_H^{cart}(0) = (40,19,110), \mathbf{v}_H(0) = (0.5,0)$ $\mathbf{p}_I^{cart}(0) = (0,20,-50), \mathbf{v}_I(0) = (0.5,-1.2)$ $\mathbf{p}_J^{cart}(0) = (30,10,-150), \mathbf{v}_J(0) = (0.5,0)$ $\mathbf{p}_K^{cart}(0) = (14,14,-20), \mathbf{v}_K(0) = (0.5,0)$

The simulations are conducted using the initial conditions and the obstacle information as presented in Table 6.4. The obstacles have the different linear velocities and non-zero angular velocities in scenario 3. The number of obstacles and the simulation area increased considerably in scenario 4. The

robot travels to the destination avoiding ten obstacles using the collision avoidance algorithms in order to verify the performances of the algorithms in the complex environments.

Two more scenarios are simulated for specific cases. The obstacles, which are positioned in a circle, move to the center in scenario 5. In scenario 6, the obstacles change their velocity suddenly on their way. The initial conditions and the obstacle information of the two scenarios are as presented in Table 6.5.

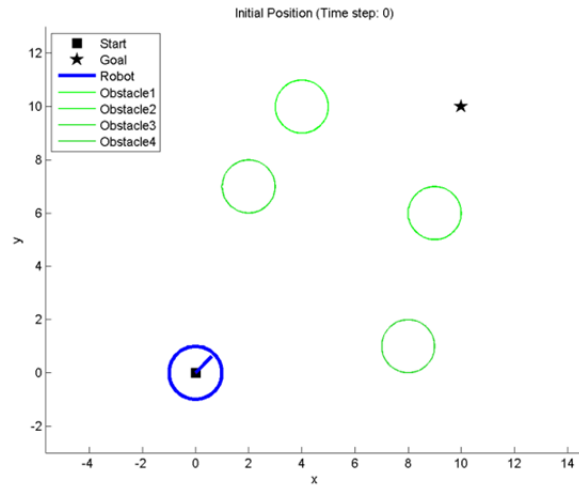
TABLE 6.5  
INITIAL INFORMATION FOR SCENARIO 5 AND 6

Initial information (Common)	Robot position and orientation	$\mathbf{p}_A^{cart}(0) = (x_A(0), y_A(0), \theta_A(0)) = (0, 0, 45)$
	Robot velocity	$\mathbf{v}_A(0) = (\dot{r}_A(0), \dot{\theta}_A(0)) = (1, 0)$
	Goal position	$\mathbf{p}_{goal}^{cart}(0) = (x_{goal}, y_{goal}) = (30, 30)$
Scenario 5	Obstacle information	$\mathbf{p}_B^{cart}(0) = (30, 30, -135), \mathbf{v}_B(0) = (1, 0)$ $\mathbf{p}_C^{cart}(0) = (15, 36.2, -90), \mathbf{v}_C(0) = (1, 0)$ $\mathbf{p}_D^{cart}(0) = (0, 30, -45), \mathbf{v}_D(0) = (1, 0)$ $\mathbf{p}_E^{cart}(0) = (-6.2, 15, 0), \mathbf{v}_E(0) = (1, 0)$ $\mathbf{p}_F^{cart}(0) = (15, -6.2, 90), \mathbf{v}_F(0) = (1, 0)$ $\mathbf{p}_G^{cart}(0) = (30, 0, 135), \mathbf{v}_G(0) = (1, 0)$ $\mathbf{p}_H^{cart}(0) = (36.2, 15, 180), \mathbf{v}_H(0) = (1, 0)$
Scenario 6	Obstacle information	$\mathbf{p}_B^{cart}(0) = (30, 5, 100), \mathbf{v}_B(0) = (0.9, 3)$ $\mathbf{p}_C^{cart}(0) = (20, 10, 130), \mathbf{v}_C(0) = (0.7, -2)$ $\mathbf{p}_D^{cart}(0) = (15, 30, -45), \mathbf{v}_D(0) = (0.5, 1)$ $\mathbf{p}_E^{cart}(0) = (5, 20, 10), \mathbf{v}_E(0) = (1.1, -1)$ $\mathbf{p}_F^{cart}(0) = (6, 15, -30), \mathbf{v}_F(0) = (0.7, 0)$

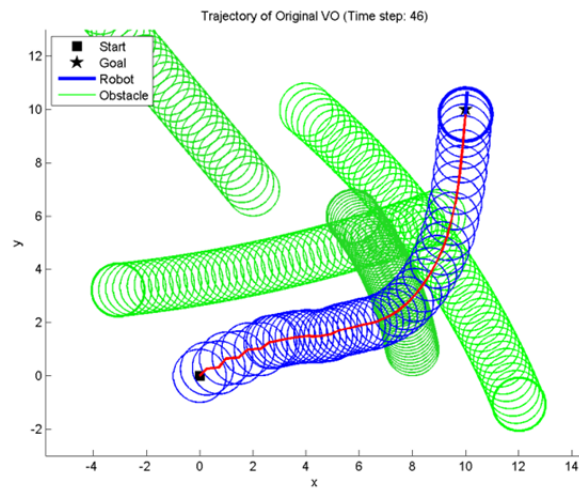
### 6.3.1 Scenario 3: Four Moving Obstacles

Figure 6.9 shows traces of the robot and the four obstacles in scenario 3 for three variations of the velocity obstacle algorithms. When the robot uses the original velocity obstacles and the dynamic velocity space, the trajectories are not smooth due to oscillations. Furthermore, the robot with the original velocity obstacles moves with very slow linear velocity due to the stuck situation, and the dynamic velocity space cannot avoid the collisions well. The robot using the original algorithm is stuck as shown in Figure 6.10 (a) almost more than half of the total time.

The proposed algorithm brings smooth and safe movement of the robot as shown in Figure 6.9 (d) and Table 6.6. The dynamic velocity space did not guarantee collision-free from multiple obstacles. When the obstacle is very close to the robot, the velocity space is almost occupied by the collision regions. In this situation, the robot has to stop by the dynamic velocity space algorithm. Furthermore, geometric continuity rate of the proposed algorithm is superior to the other algorithms, because the proposed algorithm tries to maintain the curvature of the trajectory even interrupted by the obstacles. The other algorithms do not care the smoothness of the trajectory. Geometric continuity rate of the proposed algorithm is 74.39%, and the rate is about twice geometric rate of the other algorithms.

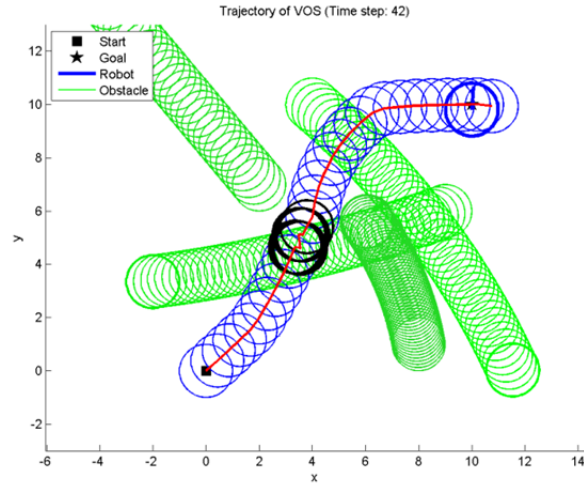


(a) Initial position

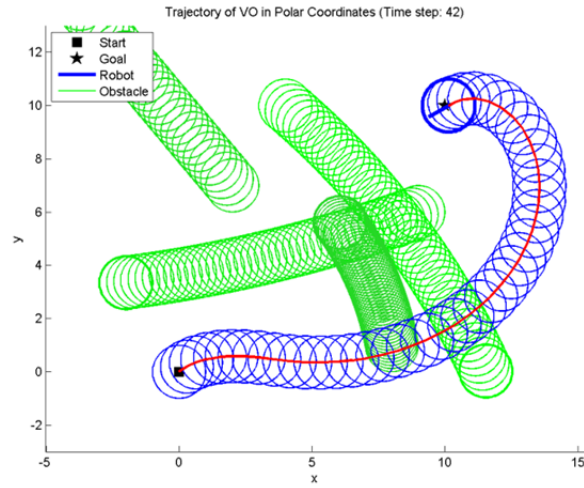


(b) Trajectory of VO



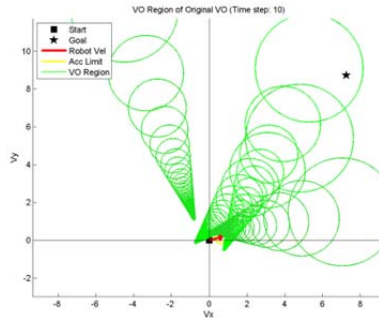


(c) Trajectory of DVS

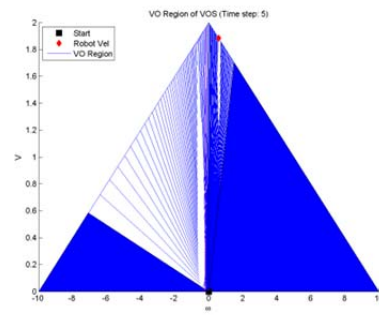


(d) Trajectory of VOP

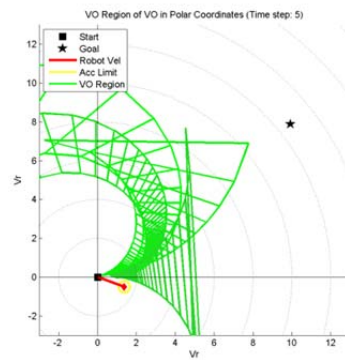
Figure 6.9 The global map and the robot trajectories of the third scenario with four moving obstacles with non-zero angular velocities. The robot using dynamic velocity space collides with the obstacle several times. The robot using the proposed algorithm has the longest path, but it arrives at the goal first.



(a) Velocity obstacle regions of VO



(b) Velocity obstacle regions of DVS



(c) Velocity obstacle regions of VOP

Figure 6.10 The velocity obstacle regions of the three collision avoidance algorithms of the third scenario at the initial part. The robot using the original velocity obstacles is in the stuck situation.

TABLE 6.6  
COMPARISON OF THE SIMULATION RESULTS OF SCENARIO 3

Algorithm	Total traveling time (s)	Total traveling distance (m)	Computation time per step (s)	Collisions (number)	Geometric continuity rate (%)
VO	13.5	16.17	0.0123	0	31.11
DVS	12.3	16.62	0.0501	11	39.02
VOP	12.3	22.66	0.0797	0	74.39

However, the robot using the proposed algorithm has the longest path among the three collision avoidance algorithms. Because the velocity obstacle regions converged on the left side of the polar coordinates as in Figure 6.10 (c), the robot had to move along the right end boundary of the velocity obstacle regions. However, the robot using the proposed navigation algorithm arrives at the goal in the fastest time. It means that the proposed algorithm generates the largest linear velocity on average.

If the time parameter of the velocity obstacle region generation process ( $\kappa$  of the equation (4.12)) is small, the detour path can be diminished. However, when several obstacles exist between the robot and the goal, small  $\kappa$  causes frequent change of the speed and the orientation.

The computation time of the dynamic velocity space and the proposed

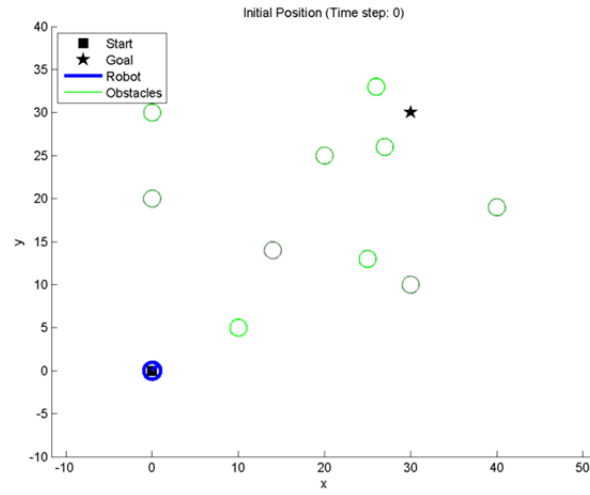
algorithm took longer than the previous simulations, but the original velocity obstacle algorithm calculates collision avoidance motion in shorter time before. It can be proven that generating discs and calculating their occupancies do not take much time. However, as confirmed before, the original velocity obstacle algorithm does not maintain high levels of the performance evaluation indices for robot navigation. If the computation time does not cause severe problems due to computational load, safety and the evaluation indices of robot path should have a priority over other performance indices.

### **6.3.2 Scenario 4: Ten Moving Obstacles**

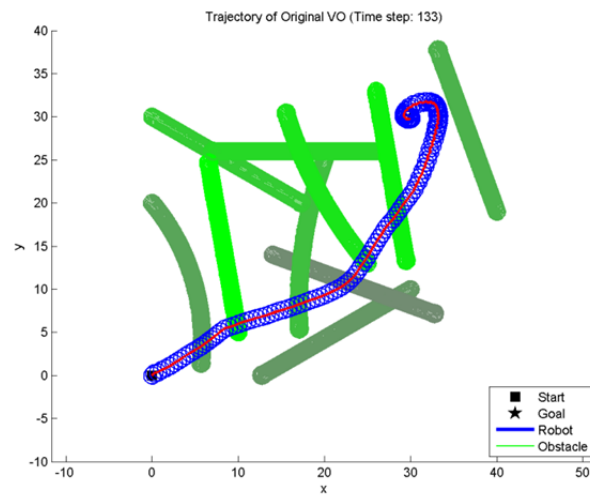
The number of the obstacles considerably increases in scenario 4. The simulation space is also enlarged as 30 by 30 meters. Because the environment becomes more complicated than the previous scenarios, the shapes of the robot trajectories of the three algorithms are quite different as shown in Figure 6.11.

The original velocity obstacle algorithm frequently corrects the collision avoidance result and changes the robot direction. Consequently, geometric continuity rate of the path is extremely low (28.79%) as shown in Table 6.7. Furthermore, the robot wanders around the goal due to the obstacle H. Because the obstacle H is passing close to the destination about  $t=30\text{sec}$ , the robot with the original velocity obstacles is directly affected by the velocity

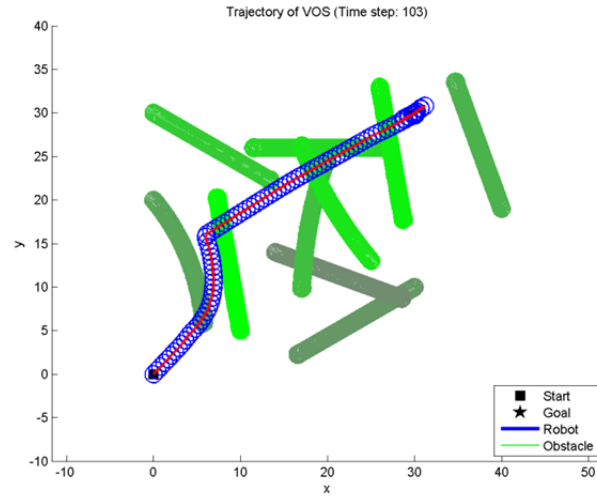
obstacle region of the obstacle H at that moment.



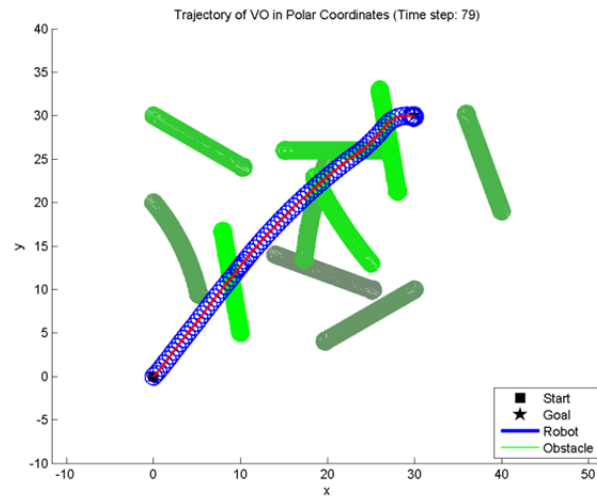
(a) Initial position



(b) Trajectory of VO

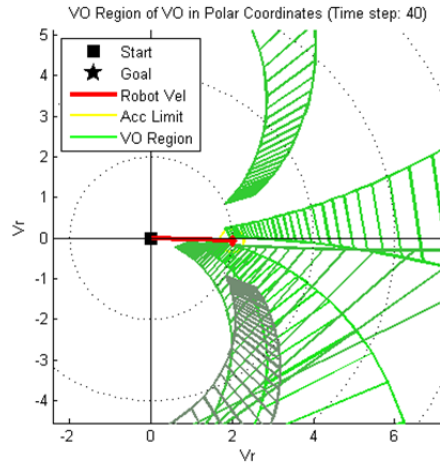


(c) Trajectory of DVS

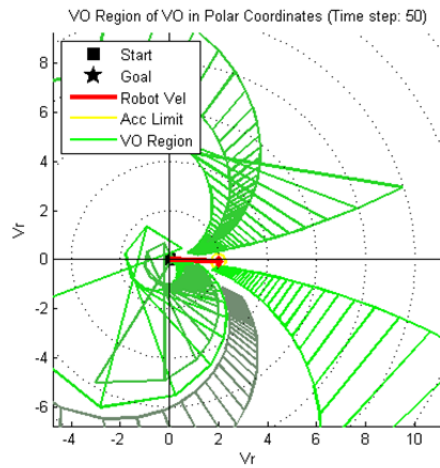


(d) Trajectory of VOP

Figure 6.11 The global map and the robot trajectories of the fourth scenario with ten moving obstacles with non-zero angular velocities. The proposed algorithm guides the robot to the shortest and fastest way compared to the conventional algorithms.



(a) Velocity obstacle regions of VOP at  $t=12\text{sec}$



(b) Velocity obstacle regions of VOP at  $t=15\text{sec}$

Figure 6.12 The velocity obstacle regions of the proposed navigation algorithm. The horizontal axis is very crowded with the velocity obstacle regions, but the proposed algorithm finds the appropriate velocity vector as new velocity of the robot.

TABLE 6.7  
COMPARISON OF THE SIMULATION RESULTS OF SCENARIO 4

Algorithm	Total traveling time (s)	Total traveling distance (m)	Computation time per step (s)	Collisions (number)	Geometric continuity rate (%)
VO	39.6	56.73	0.0135	0	28.79
DVS	30.6	52.14	0.0487	0	56.86
VOP	23.4	42.99	0.0786	0	77.56

The robot using the dynamic velocity space algorithm moves parallel to the obstacle B at the early time of the simulation. After passing the front of the obstacle B, the robot turns to the destination sharply. The robot moves straight forward after turning motion. However, geometric continuity is low as 56.86%, since the robot repetitively corrects its direction when it avoids the obstacle B.

The proposed velocity obstacle algorithm generates the shortest and fastest trajectory in this scenario, since the proposed algorithm calculates proper results in crowded situation. As shown in Figure 6.12, the robot senses more than five obstacles at once. When the original velocity obstacle algorithm faces the same situation, we can imagine that the velocity obstacle regions would occupy the horizontal axis of the velocity coordinates since the velocity obstacle regions are cone-shaped region. However, the proposed



algorithm generates the curved velocity obstacle regions, and it can be calculated how the velocity obstacle region bends. In this scenario, the proposed algorithm does not generate the detour paths. Geometric continuity rate of the proposed algorithm is also superior to the others. Therefore, it is shown that the velocity obstacles in polar coordinates can generate the robot path which shows outstanding performance evaluation indices in comparison with other conventional techniques in the large complex space.

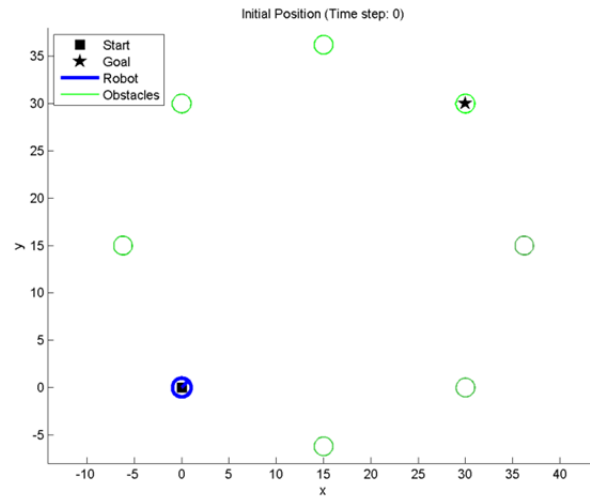
### **6.3.3 Scenario 5: Seven Moving Obstacles in a Circle**

In this scenario, initial positions of seven moving obstacles and the robot form a circle. The obstacles move toward to the center of the circle, and the robot moves to the opposite position of the circle. Figure 6.13 (a) shows the initial positions of the robot and obstacles.

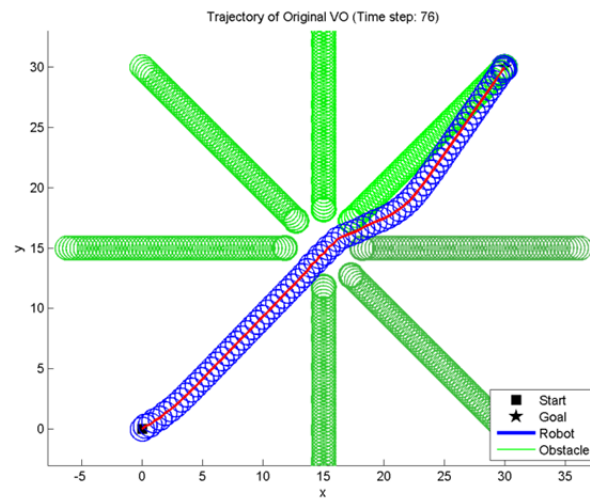
As presented in Figure 6.13 (b) through (d) and Table 6.8, three variations of the velocity obstacle algorithms generate robot trajectories whose travel length and time are similar. However, the geometric continuity rate of the proposed algorithm overwhelms the rate of other algorithms.

The robot using the original velocity obstacle algorithm has excessive collision avoidance motion in a short time to avoid an obstacle which starts from the opposite direction, so smoothness rate gets lower than the proposed algorithm. The robot using the dynamic velocity space also follows the trajectory with low smoothness. The dynamic velocity space algorithm

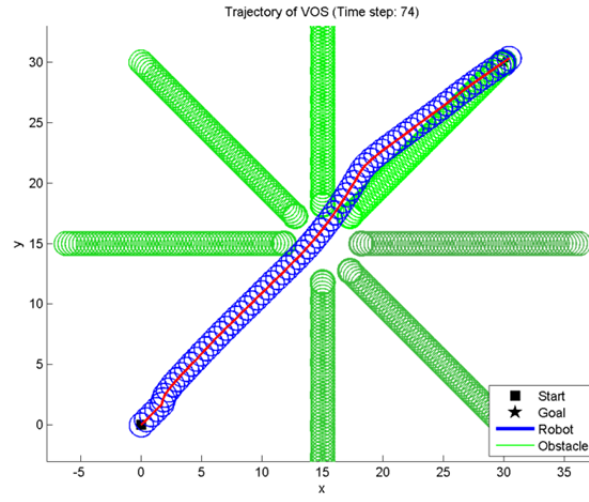
generates the sharpest robot path as presented in Figure 6.13 (c).



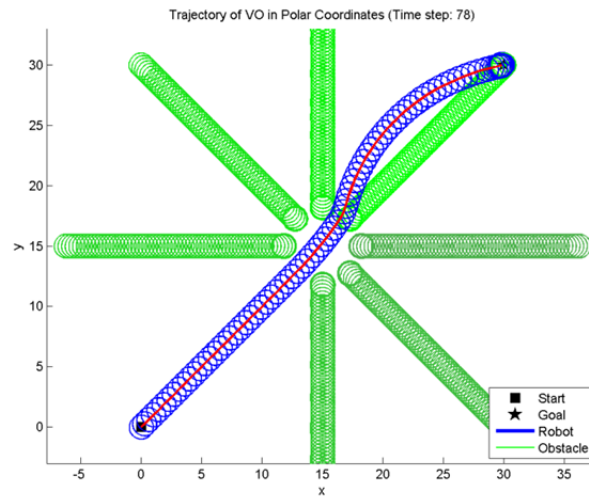
(a) Initial position



(b) Trajectory of VO



(c) Trajectory of DVS



(d) Trajectory of VOP

Figure 6.13 The global map and the robot trajectories of the fifth scenario with seven moving obstacles whose starting positions form a circle. The proposed algorithm guides the robot to the smoothest way compared to the conventional algorithms.

TABLE 6.8  
COMPARISON OF THE SIMULATION RESULTS OF SCENARIO 5

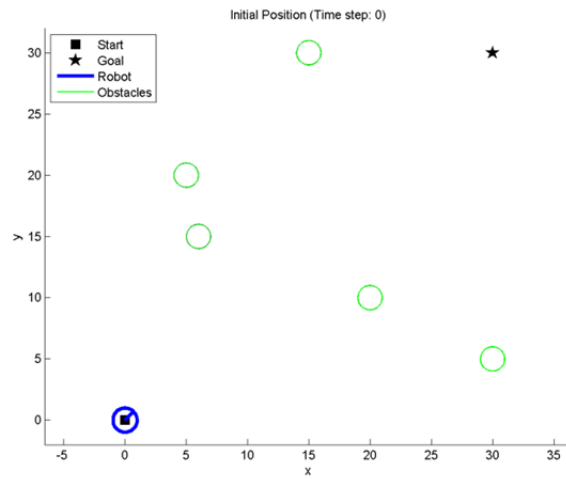
Algorithm	Total traveling time (s)	Total traveling distance (m)	Computation time per step (s)	Collisions (number)	Geometric continuity rate (%)
VO	22.5	43.04	0.0064	0	69.33
DVS	21.9	43.36	0.054	0	54.11
VOP	23.1	43.38	0.0976	0	92.21

#### 6.3.4 Scenario 6: Five Moving Obstacles with Velocity Changes

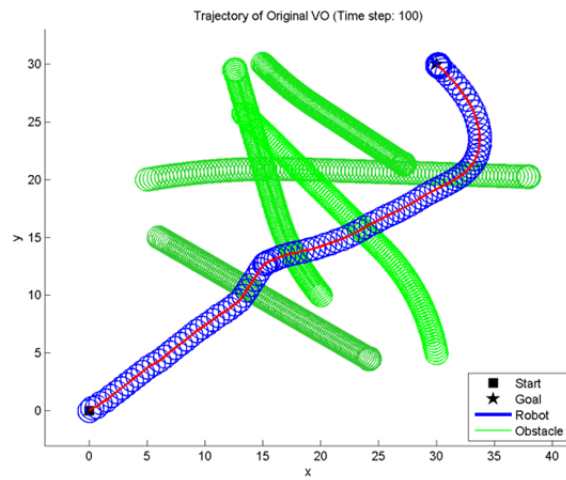
Velocity change of the obstacles is considered in scenario 6. The obstacles start their motion with initial velocities as presented in Table 6.5, and reverse their angular velocities after  $t=9\text{sec}$ . The initial positions of the robot and the obstacles are shown in Figure 6.14 (a). The obstacle F has zero angular velocity, so it travels the straight path to the end of the simulation. However, as presented in Figure 6.14 (b) through (d), curvatures of the trajectories of other obstacles are reversed after halfway.

Due to the change of angular velocities of the obstacles, the robot using the original velocity obstacles suffers severe local minima problem about  $t=12\text{sec}$ . The robot is surrounded by the obstacles and it cannot calculate new

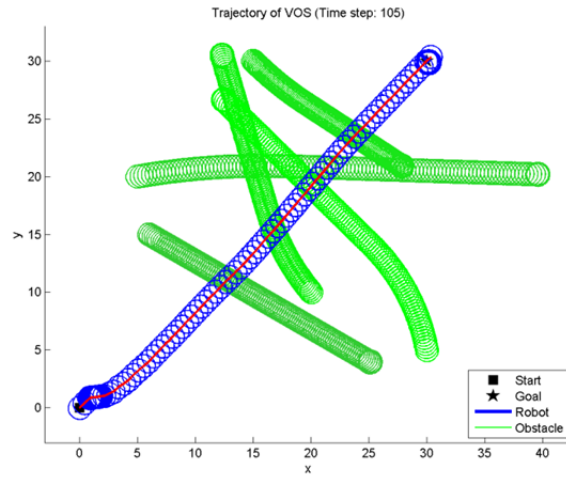
velocity to escape the situation. Finally, the robot stops between the obstacles, so the total traveling time increase and the geometric continuity rate is low compared to other algorithms.



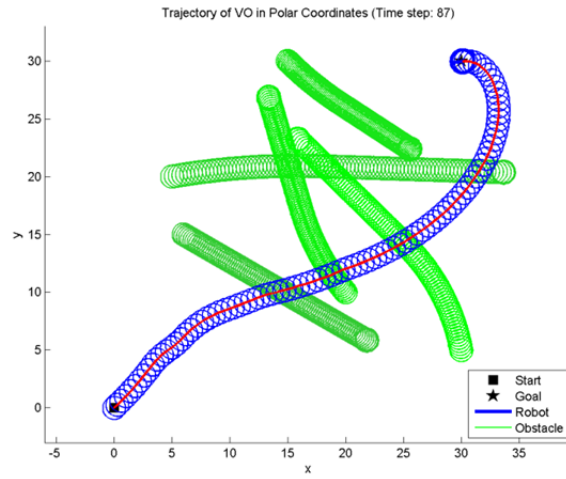
(a) Initial position



(b) Trajectory of VO



(c) Trajectory of DVS



(d) Trajectory of VOP

Figure 6.14 The global map and the robot trajectories of the sixth scenario with five moving obstacles. The angular velocities of the obstacles change after  $t=9\text{sec}$ . The proposed algorithm guides the robot to the fastest and smoothest way.

TABLE 6.9  
COMPARISON OF THE SIMULATION RESULTS OF SCENARIO 6

Algorithm	Total traveling time (s)	Total traveling distance (m)	Computation time per step (s)	Collisions (number)	Geometric continuity rate (%)
VO	29.7	50.01	0.0117	0	39.4
DVS	31.2	43.57	0.0313	0	52.4
VOP	25.8	50.06	0.0627	0	70.93

The robot using the dynamic velocity space barely moves until  $t=12\text{sec}$ . The robot travels about five meters during twelve seconds, so the total traveling time of the dynamic velocity space is the longest value among the three algorithms. However, the total traveling distance is the shortest since the robot can find the direct path to the destination after the disappearance of congestion.

The proposed algorithm guides the robot along the fastest and the smoothest trajectory. The computation time per iteration step is about double of the computation time of the dynamic velocity space. However the computation time is about one fifth of the time interval. Therefore, it is proven that most of the performance indices of the proposed algorithm are better than the conventional algorithms as shown in Table 6.9.

## **Chapter 7**

### **Conclusions**

This dissertation presents a novel mobile robot navigation algorithm for multiple obstacles avoidance in dynamic environments. The velocity obstacles approach in polar coordinates is a distributed navigation algorithm without communication with nearby robots, so the robot conducts the proposed algorithm using only sensor information from itself. The major difference from the conventional velocity obstacles is that analysis and representation of the entire obstacle avoidance procedure using the velocity obstacles concept is processed in robot-centered polar coordinates. Since obstacle velocity is considered at the velocity obstacle region generation process, the velocity obstacle region is represented in a simple form. It is also considered the dynamics and acceleration constraints of a two-wheeled differential drive mobile robot model.



The differences from the conventional algorithms guide the robot quickly to the destination and satisfying high levels of the performance evaluation indices of the robot path. The performance evaluation indices of driving are analyzed in three different terms. The new velocity obstacles approach is developed with focuses on collision-free, smoothness, and local minima avoidance. Firstly, the proposed algorithm generates collision-free paths of the robot using a velocity obstacles concept. By interpreting velocity space of the robot in robot-centered polar coordinates, the entire procedure of obstacle avoidance can be conducted on the same representation. As a result, the kinematic constraints of the robot can be easily applied to velocity calculation.

Secondly, the proposed algorithm pursues driving the robot without oscillation and sudden velocity change. Oscillation and sudden velocity can be prevented by predicting the future situation properly. The proposed algorithm predicts the obstacles' future movements using various factors such as the radii of rotation of the velocity obstacle region, the shape of the velocity obstacle regions, and so on. The proposed algorithm is designed to maintain the radius of rotation of the robot as much as possible using the factor parameterizations. Accordingly, the robot can have the smooth trajectory which satisfies geometric continuity. Decrease of sudden turning motion of the robot is verified in most of the simulations. The proposed algorithm also reduces oscillation of the robot path and avoids stuck situation, and it is verified in Section 6.

The robot can be caught on local minima occasionally as shown in the cases of the original velocity obstacles in Section 6.2.2 and 6.3.1. However, the proposed algorithm avoids local minima and generates a smooth path in the same situation. Moreover, it serves a faster path than the conventional technique to the robot.

The effects of analyzing the velocity obstacles concept in the robot-centered polar coordinate system are evaluated through various simulations. Most of the results of the proposed algorithm were better than the conventional algorithms as shown in Table 7.1. In most of the simulation scenario, the robot using the proposed navigation algorithm is the first to arrive at the destination even though the robot travels the longest path. The computation time per iteration of the proposed algorithm is the longest among the three collision avoidance algorithms. However, the average computation time of the proposed algorithm is about 20% of the time interval, so it does not affect collision avoidance motion of the robot at all. The original velocity obstacle algorithm has the average computation time as 0.012 seconds, but it takes the total traveling time more than 4 seconds compared with the proposed algorithm. The original velocity obstacles and the proposed algorithm generate collision-free path of the robot although under complex and crowded conditions. On the other hand, the dynamic velocity space cannot achieve collision-free navigation if angular velocity of the obstacle is large. Therefore, it is verified that the dynamic velocity space is the most vulnerable to

rotational motion of the obstacle among the three collision avoidance techniques. Smoothness, which signifies the performance evaluation of the path, of the proposed algorithm records 79.03% geometric continuity, and it is higher than two times as geometric continuity of the original velocity obstacles.

TABLE 7.1  
OVERALL COMPARISON OF THE SIMULATION RESULTS

Algorithm	Average time difference from the fastest case (s)	Average computation time (s)	Average collisions (number)	Average geometric continuity rate (%)
VO	4.4	0.012	0	35.75
DVS	2.2	0.044	2.17	53.11
VOP	0.2	0.066	0	79.03

However, the proposed algorithm still has several issues to compliment. Usually, the velocity obstacle regions are gathered together, and they are located near the  $\dot{\theta}_A = 0$  axis of the robot-centered polar coordinates. Therefore, the robot prevents to be caught in local minima, but the robot may lose chances to take a shortcut avoiding a bunch of the velocity obstacle regions. Secondly, when the obstacles are located on both sides of the robot and they are not approaching to the robot, the robot may not move to the

forward direction. Because the proposed algorithm considers only the alternative vectors from the left and right half planes. This case can be reduced by considering another alternative vectors from the front region of the robot. In the future, the proposed algorithm would be adapted for other kinematic systems and more complex dynamic constraints. In the future, the velocity obstacles approach could be analyzed insightfully and modified to generate a shorter path than the current algorithm.

# Bibliography

- [1] J.-C. Latombe, *Robot Motion Planning*, Kluwer academic publishers, 1991.
- [2] K. Azarm and G. Schmidt, “Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3526-3533, 1997.
- [3] T.-Y. Li and H.-C. Chou, “Motion planning for a crowd of robots,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4215-4221, 2003.
- [4] S.H. Park and B.H. Lee, “Analysis of robot collision characteristics using the concept of the collision map,” *Robotica*, vol. 24, no. 3, pp. 295-303, 2006.
- [5] J.S. Choi, Y.H. Yoon, M.H. Choi, and B.H. Lee, “Parameterized collision region for centralized motion planning of multi-agents along specified paths,” *Robotica*, vol. 29, no. 7, pp. 1059-1073, 2011.

- [6] J.H. Reif, "Complexity of the mover's problem and generalizations," in *Proceedings of the 20<sup>th</sup> IEEE Symposium on Foundations of Computer Science*, pp. 412-427, 1979.
- [7] J.F. Canny, *The Complexity of Robot Motion Planning*, The MIT Press, 1998.
- [8] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pp. 624-631, 2002.
- [9] M. Turpin, N. Michael, and V. Kumar, "Trajectory planning and assignment in multirobot systems," in *Workshop on the Algorithmic Foundations of Robotics*, Boston, MA, 2012.
- [10] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," *Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics*, vol. 83, pp. 545-558, 2013.
- [11] I. Mas and C. Kitts, "Centralized and decentralized multi-robot control methods using the cluster space control framework," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 115-122, 2010.

- [12] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, pp. 760-772, 1998.
- [13] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3716-3721, 2001.
- [14] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5573-5578, 2009.
- [15] B. Kluge and E. Prassler, "Recursive probabilistic velocity obstacles for reflective navigation," in *Field and Service Robotics: Recent Advances in Research and Applications*, (*Springer Tracts in Advanced Robotics*, vol.24), S. Yuta, H. Asama, S. Thrun, E. Prassler, and T. Tsubouchi, Eds. Berlin, Germany: Springer-Verlag, pp. 71-79, 2006.
- [16] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1610-1616, 2007.

- [17] E. Owen and L. Montano, "A robocentric motion planner for dynamic environments using the velocity space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4368-4374, 2006.
- [18] R. Bis, H. Peng, and G. Ulsoy, "Velocity occupancy space: Robot navigation and moving obstacle avoidance with sensor uncertainty," in *Proceedings of the ASME Dynamic Systems and Control Conference*, pp. 363-370, 2009.
- [19] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233-255, 2002.
- [20] S.M. LaValle and J.J. Kuffner Jr., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378-400, 2001.
- [21] J. Borenstein and Y. Koren, "The vector field histogram – Fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278-288, 1991.
- [22] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23-33, 1997.



- [23] T. Fraichard and H. Asama, "Inevitable collision states – A step towards safer robots?," *Advanced Robotics*, vol. 18, no. 10, pp. 1001-1024, 2004.
- [24] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3726-3731, 2005.
- [25] T.S. Lee, G.H. Eoh, J. Kim, and B.H. Lee, "Mobile robot navigation with reactive free space estimation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4420-4425, 2010.
- [26] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 28, no. 5, pp. 562-574, 1998.
- [27] E. Prassler, J. Scholz, and P. Fiorini, "A robotic wheelchair for crowded public environments," *IEEE Robotics and Automation Magazine*, vol. 8, no. 1, pp. 38-45, 2001.
- [28] P. Fiorini and D. Botturi, "Introducing service robotics to the pharmaceutical industry," *Intelligent Service Robotics*, vol. 1, no. 4, pp. 267-280, 2008.
- [29] S.M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

- [30] B.A. Barsky, *Computer graphics and geometric modeling using beta-splines*, Springer-Verlag, Heidelberg, Germany, 1988.
- [31] B.A. Barsky and T.D. DeRose, "Geometric continuity of parametric curves: Three equivalent characterizations," *IEEE Computer Graphics and Applications*, vol. 9, no. 6, pp. 60-69, 1989.
- [32] V.J. Lumelsky and A.A. Stepanov, "Path planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 403-430, 1987.
- [33] V.J. Lumelsky, "A comparative study on the path length performance of maze-searching and robot motion planning algorithms," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 57-66, 1991.
- [34] H.P. Huang and P.C. Lee, "A real-time algorithm for obstacle avoidance of autonomous mobile robots," *Robotica*, no. 10, no. 3, pp. 217-227, 1992.
- [35] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 6, pp. 814-822, 1997.
- [36] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224-241, 1992.

- [37] S.G. Loizou and K.J. Kyriakopoulos, "Closed loop navigation for multiple holonomic vehicles," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 2861-2866, 2002.
- [38] S.G. Loizou and K.J. Kyriakopoulos, "Navigation of multiple kinematically constrained robots," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 221-231, 2008.
- [39] A. Krause, A. Gupta, C. Guestrin, and J. Kleinberg, "Near-optimal sensor placements: Maximizing information while minimizing communication cost," in *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, pp. 2-10, 2006.
- [40] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Robust sensor placements at informative and communication-efficient locations," *ACM Transactions on Sensor Networks*, vol. 7, no. 4, art. no. 31, 2011.
- [41] T.P. Lambrou and C.G. Panayiotou, "Collaborative path planning for event search and exploration in mixed sensor networks," *International Journal of Robotics Research*, vol. 32, no. 12, pp. 1424-1437, 2013.
- [42] D. Keymeulen and J. Decuyper, "Fluid dynamics applied to mobile robot motion: The stream field method," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 378-385, 1994.

- [43] A.M. Hussein and A. Elnagar, "Motion planning using maxwell's equations," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 2347-2352, 2002.
- [44] S. Bandi and D. Thalmann, "Path finding for human motion in virtual environments," *Computational Geometry: Theory and Applications*, vol. 15, pp. 103-127, 2000.
- [45] P. Chaudhuri, R. Khandekar, D. Sethi, and P. Kalra, "An efficient central path algorithm for virtual navigation," in *Proceedings of the Computer Graphics International Conference*, pp. 188-195, 2004.

## 초 록

미지의 환경에서 자율주행을 하기 위하여 로봇은 정확하게 환경을 인지하고 충돌 없이 빠르게 움직이는 경로를 생성해야 한다. 최근에 운영 환경이 복잡해짐에 따라 다수의 개체와 이동 장애물과 같은 다양한 요소를 고려하는 것이 자율주행의 중요한 문제가 되었다. 따라서 다양한 상황에서 효과적으로 작동하는 충돌 회피 주행 방법을 개발하는 것이 필요하다.

중앙식 주행 시스템은 환경 정보와 모든 로봇에 대한 정보를 수집하여 각 로봇의 주행 경로를 결정한다. 따라서 운영 환경이 복잡해지면 각 로봇의 무충돌 경로를 계산하는 작업은 어려워진다. 각 로봇 별로 제어하는 분산식 주행 시스템은 로봇의 최적 경로를 보장할 수 없지만, 상황에 따라 적용하기가 용이하다. 본 학위 논문은 중앙식 시스템의 제어 방식을 사용하지 않는 지역적, 반응적 주행 기법에 대해 다루었다.

본 논문에서는 지역적, 반응적 주행 기법 중 하나인 velocity obstacle 개념을 재해석하였다. 대부분의 기존의 velocity obstacle 기법들은 Cartesian 좌표계를 기반으로 하여 사용된다. 본 연구에서

는 각속도 성분을 가지는 로봇과 장애물의 충돌을 예상하고 충돌 회피 움직임을 계산하는 작업을 로봇 중심의 극 좌표계를 기반으로 하여 수행한다. Velocity obstacles 개념을 로봇 중심의 극 좌표계에서 재해석함으로써 장애물 회피 과정은 간단하게 표현된다.

기존의 velocity obstacle 방법들을 사용하면 로봇과 이동 장애물의 이동 방향에 따라 로봇은 진동하는 움직임을 가지게 된다. 진동으로 인해 발생하는 문제를 극복하기 위해서 무진동 경로를 유지하며 충돌을 회피할 수 있는 로봇의 속도를 생성하는 새로운 방법이 제시되었다. 로봇의 현재 상태, 장애물과의 상대 관계에 대한 정보, 목적지까지의 거리 등을 포함한 비용 함수가 제시되어 로봇의 속도를 결정하는 데 사용하였다.

다양한 시뮬레이션을 통해 제시한 방법과 기존의 충돌 회피 방법들의 성능을 검증하였다. 기존의 충돌 회피 방법과 주행 시간, 주행 거리, 연산 시간, 경로의 부드러움 등을 비교하여 제시한 알고리즘이 더 좋은 충돌 회피 성능을 나타냄을 보였다.

**주요어:** 충돌 회피, Velocity obstacles, 극 좌표계, 다개체 시스템, 운동 계획법

**학 번:** 2009-30205

## 감사의 글

많은 분들의 도움으로 본 박사 논문을 작성할 수 있었고 대학원에서 연구 생활을 무사히 마칠 수 있었기에, 소중한 분들에게 이 글을 통해 감사하는 마음을 전합니다.

지난 7 년간 많은 가르침을 주신 지도교수님이신 이범희 교수님께 진심으로 감사 드립니다. 교수님께서 제게 보여주신 믿음과 후원, 말씀해주신 칭찬과 조언 덕분에 연구와 논문을 완성할 수 있었습니다. 그리고 교수님께 배운 삶의 자세를 졸업 후에도 잊지 않겠습니다. 감사합니다.

그리고 연구와 강의로 바쁘신 중에도 귀한 시간을 내어 이 학위 논문을 심사해주신 교수님들께도 감사 드립니다. 심사위원장으로 모신 최진영 교수님, 전체적인 부분부터 세부적인 부분까지 이 논문에 대해 많이 조언해 주셔서 감사합니다. 인자하시고 편안하신 교수님의 성품과 연구에 대한 열정을 본받도록 하겠습니다. 열정적인 모습으로 많은 학생들에게 연구의욕을 불어넣어주시는 오성희 교수님께도 감사 드립니다. 기술적인 내용을 짚어주시고 이 연구의 나아갈 방향에 대해서 심도 있게 조언해주셔서 감사합니다. 제가 미처 생각하지 못했던 접근 방향을 말씀해주신 이동준 교수님, 교수님의 통찰력과 조언으로 이 연구의 부족한 부분을 채울 수 있었습니다. 감사합니다. 먼 곳에서 오시기 불편함에도 불구하고 흔쾌히 심사에 응해주신 박재병 교수님께도 진심으로 감사 드립니다. 제 연구와 생활에 많은 영향을 주신 소중한 분 이십니다.

석사, 박사 과정을 함께 보내고 서로 의지했던 많은 선후배님들께도 진심으로 감사 드립니다. 프로젝트 회의에서 자주 뵈는 상훈이형, 항상 기운을 북돋아 주셔서 감사합니다. 석사시절

흐트러짐 없는 생활을 하게 도와주신 연수형께도 감사합니다. 항상 진취적으로 도전하시는 금배형, 밤늦게 있을 때 응원해주셔서 감사합니다. 바람직한 삶의 자세란 무엇인지 생각하게 늘 귀감이 되시는 정식이형, 깊은 감사의 마음을 전해드립니다. 늘 힘을 주셔서 고맙습니다. 학부 졸업 프로젝트부터 긴 시간 동안 도움주신 공우형, 항상 감사합니다. 서윤이와 형수님과 행복하세요. 활기찬 모습으로 넘치는 에너지를 보여주신 부산 사나이 정희형, 연구를 위해 헤어스타일 정도는 쉽게 바꾸시는 노산이형, 덕분에 연구실 생활이 즐거웠습니다. 진중하면서도 매력적인 모습을 갖고 계신 영환이형과의 연구실 생활도 즐거웠습니다. 모두 감사합니다. 이제는 미국에서 신혼과 학업을 병행하시는 우현이형과 신규형에게도 응원의 메시지를 보냅니다. 꿈꿈함과 따뜻한 감성을 가진 완벽한 신여성 지민누나, 연구실 생활 편하도록 신경 써주셔서 고맙습니다. 따뜻한 캘리포니아의 햇살을 맞으며 유학중인 인규형도 힘내시라고 응원합니다. 한국 유학시절 남들 모르게 힘들었을 천슈형과 헤이룽에게도 고마운 마음을 전합니다. 석사 동기이자 1 월 레바논 파병을 앞두고 있는 우종석 소령님, 새로운 도전이 승승장구하는 발판이 되고 소중한 경험이 되기를 바랍니다. 출디추운 보스턴의 혹한을 견디며 공부하고 있는 중희도 힘든 유학생활을 서영씨와 함께 행복하게 완성해 나가기를 바랍니다. 신혼의 달콤함을 만끽하고 있는 준석이형, 앞으로도 좋은 일만 가득하기를 바라겠습니다. 이젠 댄디한 직장인의 모습을 물씬 풍기는 경희와 진수도 목표한 바 다 이루고, 곧 좋은 소식 들려주면 좋겠습니다. 고국에서의 근무를 끝내고 화려하게 서울로 컴백한 야니스, 종종 술잔을 기울이자. 얼마 전, 영국에서 박사과정을 시작한 니콜라이도 힘내라고 응원합니다. 같이 놀던 게 엇그제 같은 승희와 경식이도 이젠 사회인으로서 새로운 도전들을 하게 될 텐데 항상 좋은 성과 있기를 바랍니다. 모두 감사합니다.

바로 다음학기에 졸업 준비중인 두진형, 형의 능력이라면 충분하다고 생각해요. 곧 축하의 말을 전할 수 있을 것이라



믿습니다. 누구 못지않게 긴 시간 술잔과 인생과 연구를 함께한 승환이, 연구실의 새로운 황금시대가 너의 손에 달려있다, 파이팅! 이제는 방장이 잘 어울리는 규호, 그는 나에게 많은 충격을 안겨주었습니다. 핵심 인재로서 고된 일은 잘 이겨내고, 좋은 일이 가득하길 바랍니다. 이제 본격적인 연구 생활을 시작하게 될 재도에게도 응원의 메시지를 보냅니다. 혼자 고민하고 있을 때 덕분에 많은 난관들이 해결되었습니다. 고맙습니다. 곧 빛을 볼 귀여운 삐약이와 연구와 회사 일로 정신 없을 훈수형에게도 힘내시라고 말하고 싶습니다. 행복하세요. 박사라는 길고 긴 경주의 출발 선에 선 정현이와 원석이, 중간에 넘어지지 않게 단단히 준비하여 완주하는 영광의 순간을 금방 맞이하기를 바랍니다. 정현이는 규호의 개그를, 원석이는 원거리 거주자를 조심하기를 당부합니다. 입학하기 전부터 유독 고생도 많이 하고 고민도 많이 한 고무고무 현기와 귀요미 지웅이도 늘 응원하고 있습니다. 위낙 능력이 출중하기 때문에 잘 이겨냈으니, 내실을 다지게 된 기회라고 생각하면 좋겠습니다. 로맨틱 기타리스트에서 우주갑부까지 거듭나게 된 지훈이도 즐거운 연구실 생활 이어가기를 바랍니다. 이제 막 연구실 생활을 시작한 현우와 원영이도 목표한 바 성취하고 뜻 깊은 대학원 과정을 보내기를 바랍니다.

제 곁에서 항상 응원해주는 많은 분들께도 감사 말씀을 드립니다. 룰루반 03 친구들, 부천고와 희망학원 친구들, 현대차 장학생 친구들, 그리고 그 외에도 격려해주고 신경 써주신 많은 분들, 고맙습니다.

마지막으로 30 년간 제일 고생 많으셨던 가족 분들께 진심으로 감사합니다. 헌신적인 모습으로 항상 사랑해주신 엄마, 아빠, 고맙습니다. 두 분의 크신 사랑을 기억하고, 보답하며, 베풀고 살겠습니다. 가정을 꾸린 후에도 자주 신경 써주는 누나와 매형, 그리고 점점 의젓해질 도윤이, 건강하시길 바라는 할머니까지 모두에게 감사한 마음을 전합니다. 이 모든 기쁨과 영광을 가족에게 바칩니다.